# Secure multiparty computation on incomplete networks

SHAILESH VAYA*.

Department of Computer Science and Engineering,
Indian Institute of Technology Madras,
India - 600036

* Most of this work was done at the Department of Computer Science and Engineering, Indian Institute of Technology Madras; Contact email:vaya@cse.iitm.ernet.in

**Abstract.** Secure multiparty computation of a multivariate function is a central problem in cryptography. It is known that secure multiparty computation can be realized by a set of *n* parties iff the connectivity of the underlying (authenticated) communication network is more than twice the number of corrupted parties. This impossibility result makes secure multiparty computation far less applicable in practice, as most deployed networks have a much lower degree than $O(n)$ and one would ideally like to tolerate $\theta(n)$ corrupted parties.
This work considers a model for (unconditional) secure multiparty computation for networks of low degrees in which authenticated channels are available between very few pairs of parties. Not all honest parties can achieve traditional security guarantees of multiparty computation for this setting. This formulation of secure multiparty computation, which permits some of the honest parties to be "sacrificed" is called almost everywhere secure computation. In this work we show how to realize a.e.s.c., on a few special families of incomplete networks, for the case of Byzantine corruptions.

Keyword(s): secure multiparty computation, Byzantine agreement, almost everywhere agreement, almost everywhere secure computation, bounded degree networks, fault tolerant networks.

## 1 Introduction

In secure multiparty computation *n* players jointly evaluate an arbitrary *n*-variate polynomial time computable function $f$ on the vector of their input values. The strong guarantee for secure MPC is that even if a fraction of the nodes are controlled by a malicious adversary, all the honest parties should still obtain their correct output values, without leaking any more information about their individual inputs than what is revealed by the output values of the corrupted parties. Solutions for this problem were presented in [Yao82] for $n = 2$, followed by [GMW87] for *n* parties.

In [BGW88] and [CCD88], it is shown how to achieve secure multiparty computation against a computationally unbounded adversary, as long as as least $\lfloor \frac{2*n}{3} \rfloor + 1$ of the participating parties are guaranteed to be honest. This result assumes the existence of point-to-point authentication channels between every pair of parties. Since *reliable* let alone *private* channels are too expensive to realize in general, it is infeasible to assume networks with direct and private channels between every pair of parties. It is natural to wonder about the possibility of achieving the guarantees of secure multiparty computation for networks of low degree.

The above question was posed first by [Dol83], and explored further by [DDWY93] who showed that if there can be at most *t* corrupted players, then connectivity of $2*t+1$ is both necessary and sufficient to achieve information theoretically secure multiparty computation. Since one would like to be able to tolerate as large a fraction of corrupted parties as possible (say $\Omega(n)$) for networks of degree/connectivity as low as possible (say $O(1)$) - can one still obtain meaningful and useful *intermediate* results?

Since the results of [DDWY93] rule out achieving the strong guarantees of information theoretic secure multiparty computation or even Byzantine Agreement, [LSP82], [DPPU88] consider the following weaker form of agreement for networks of low degree: *An honest party could be surrounded by a set of corrupted parties. Such an honest party cannot communicate reliably with the rest of the honest parties, so achieving byzantine agreement on such networks is not possible. Can we still make any meaningful guarantees about agreement? What about the possibility of most - but not necessarily all - of the honest parties agreeing on one single value?*. In [DPPU88], the authors propose a relaxation of the agreement problem in which some of the honest parties may not be able to reach the agreement value. This type of agreement is called *almost everywhere agreement*. In [DPPU88], the following definition is proposed for an agreement protocol: A $g(n,t)$-agreement protocol, for some function $g(.,.)$, is one whose execution results in all, but $g(n,t)$ of the honest parties reaching the agreement value, when at most *t* parties are corrupted by the adversary. The authors go on to present protocols that achieve $g(n,t)$-agreement for several different families of networks.

A more generic notion then almost everywhere agreement, relevant to the setting of incomplete networks, is *almost everywhere secure computation*. This notion of MPC allows a subset of the honest parties - called sacrificed parties - to not necessarily achieve the canonical guarantees of secure multiparty computation. We distinguish between honest parties which have no hope of achieving the guarantees of secure multiparty computation because of the surrounding bad neighborhood, from the parties for which we do not

claim the canonical guarantees of secure multiparty computation. The former are called "doomed", whereas the latter also (obviously) include the former are called "sacrificed" parties. The rest of the (non-sacrificed) honest parties, which are at least $\lfloor \frac{2*n}{3} \rfloor + 1$ in number, are required to achieve the canonical guarantees of secure multiparty computation. Previous work on almost everywhere secure computation give results for the case of passive corruptions. In this work, we show how to realize almost everywhere secure computation, on a few special families of networks, handling the case of Byzantine corruptions. We also argue the tightness of the results achieved by us, in many respects.

### 1.1 Related works

This work heavily uses the infrastructure developed in the seminal work on *almost everywhere secure agreement* by [DPPU88]. Efficient protocols for this goal were presented in [BG89], [BG90]. The protocol in [Upf92], realizes almost everywhere agreement on constant degree LPS expanders, while tolerating a linear number of faults. This protocol has a polynomial round complexity, but each party executes a super-polynomial time exhaustive search. The almost everywhere agreement protocol in [OR96] is randomized. It tolerates a linear fraction of randomly located faulty processors, that fail independently with constant probability.

The notion of almost everywhere secure computation for incomplete networks, and an overall approach to realize it, was published in [GO08], by Garay and Ostrovsky. The authors present Input indistinguishability type definitions for almost everywhere secure computation. This definitional approach was first proposed in [KKMO94] handling *passive corruptions*. A hybrid argument was presented in [GO08] to reduce the privacy of multi-party computation protocol to privacy of perfect secure message transmission channels.

A new model for (unconditional) secure multiparty computation with man-in-the-middle attacks was proposed in [Vay08]. For this model of secure multiparty computation results were presented in the framework of almost everywhere secure computation. We use the definitional framework in [Vay08] to realize almost everywhere secure computation on incomplete networks for the case of active corruptions, resolving the main open problem in this line of research.

Assuming that more than $\frac{2}{3}$ parties are honest, it has been shown that it is possible to securely compute any $n$-variate function, [BGW88], [CCD88], [RBO89] in the information theoretic regime. In the computational model, the results have been given in [GMW87], [Yao82]. Adversary structures extensively used in this work, were proposed and studied in [FHM99], [HM97], [HM00] and [HMP00].

### 1.2 Organization of the paper

Section 2 discusses technical subtleties encountered in realizing almost everywhere secure multiparty computation. Section 3 is devoted to terminologies and definitions. Semi-formal definitions of secure multiparty computation and $\mathcal{T}$-secure computation are presented in subsection 3.2. Section 4, is devoted to semi-formal definitions and protocols for setting up the communication infrastructure on the incomplete network. The ancillary results are put together to prove the main theorem in this work namely, $\mathcal{T}$-secure multiparty computation on incomplete networks. We present a brief overview of the proof of the main theorem in Section 5. Finally, we present claims for realizing $\mathcal{T}$-secure computation on a few families of incomplete networks in Section 6.

## 2 Technical subtleties in realizing $\mathcal{T}$-secure computation on incomplete networks

In this section we discuss technical subtleties encountered in realizing almost everywhere secure multi-party computation on incomplete networks. The main protocol for almost everywhere secure computation is obtained by composing a standard protocol for secure multiparty computation, for the information theoretic regime (like for example [BGW88] or [CCD88]) with several (roughly $O(n^2)$ in number) message transmission protocols for sending messages between distant nodes of the incomplete networks. These protocols for

message transmission are realized using the infrastructure of the incomplete network. These protocols may not necessarily be unique and quite likely vary in the number of rounds etc. because of the asymmetry of the underlying incomplete network. We present an overview of the technicalities encountered in realizing almost everywhere secure computation for the stand alone model, for the case Byzantine adversary. We also argue the tightness of our results presented in this work.

### 2.1 Why the strongest results we hope to achieve are for the stand alone model?

First we shall argue why we are not able to use standard composition theorems for this model of (unconditional) secure multiparty computation. Without the composition theorems the strongest result one can hope to achieve is for the stand alone model. But without the composition theorems we have to compose the main protocol with (several) protocols for realizing secure message transmission protocols on incomplete networks, for the vanilla model. This makes the proof delicate even for the stand alone model.

We are not able to use standard composition theorems for this model because we encounter cases when only the correctness property (namely, correctness of the commitment of the input value or correctness of the output value received) or just the privacy property may be compromised for a sacrificed honest party. Thus the adversary does not get full control over such sacrificed parties. Furthermore, the properties that are compromised and the extent to which they are compromised may depend on the dynamic choices made by the adversary during the execution of the protocol. Thus, the sacrificed honest parties cannot be guaranteed to behave either as fully honest or as corrupted. Composition theorems are statements about the joint distribution of views of the honest and corrupted parties. But the sacrificed parties may not fall into either case. Furthermore, their final status often depends on the dynamic choices made by the adversary (Note a refinement to our definition of security where sacrifice of correctness and sacrifice of privacy property is defined separately. This refinement is noted in Appendix Subsection A.5). Thus, for this model of secure computation there is no straightforward way to utilize standard composition theorems. More generally, we find that there is no clean and satisfactory way to define the interaction of sacrificed parties with the 'Ideal world functionality'. No wonder our definition, even for the stand alone model, does not adopt the traditional approach of describing an "Ideal world process" and comparing it with a "Real world process".

Now we describe the specific scenario when the above issue arises. Depending on the choice of nodes corrupted by the adversary, the secure channels established between distant nodes of the incomplete networks utilizing the public discussion channels realized in [DPPU88], [Upf92] and the topology of the underlying incomplete network, the adversary gets the following types of control over a sacrificed party: (1) Eavesdrop the input value committed to by the party (2) Manipulate the messages sent on the channel so that the input value committed to by the party is the default value "d" (3) Arbitrarily control the input value committed to by the party (4) Manipulate the input value committed to a limited degree only, as some function of the original value say. For the first three cases the adversary also manages to learn the input value committed to by the sacrificed parties. However, for scenarios of the fourth type the adversary does not get to learn the input value but is able to influence the input value committed by the sacrificed party to a limited degree (Note that the adversary just needs read access to $\lfloor \frac{n}{3} \rfloor + 1$ channels to extract the input value committed to by the party or write access to same total number of such channels, including the ones connected to corrupt parties to be able to manipulate the committed input value of the party to the default value "d"). Thus when the fourth type case occurs, the simulator simulating the view of the adversary cannot extract the actual input value committed to by the sacrificed party and causes the technical issues discussed in the above paragraph.

### 2.2 Technicalities encountered in realizing the result for the stand alone model

Our overall approach for realizing this model of (unconditional) secure multiparty computation is: (1) Construct a complete network $N_C$ from the original incomplete network by adding virtual edges. (2) Adopt a standard proof of security for the BGW protocol to suit our definition of security for this network. (3) Order all the virtual edges in this network. And, inductively replace the virtual edges with simulations of the

PSMT protocol, while proving claims of correctness and privacy of the associated multiparty protocol for each intermediate network. In realizing this approach we encounter some technical challenges. We highlight the prominent ones which influence the structure of definitions, protocols and the proofs:

**The necessity for adversary structures:** Observe that the behavior of the virtual edges is not fixed and may vary for the same topology of the underlying incomplete network. The behavior depends on the choice of subset of parties corrupted by the adversary and the actual PSMT protocol employed (Different PSMT protocols may utilize the infrastructure of the incomplete network differently). In particular, the same virtual edge may behave as a tamperable channel or as an eavesdroppable channel or as an uncorrupted authenticated channel. Even a small variation in the subset of parties corrupted by the adversary may influence a large number of virtual edges to behave differently. Thus, we are faced with exponential number of variations. Towards this end we set up adversary structures which fixes the behavior of the virtual edges for a given choice of subsets of parties corrupted by the adversary. Obviously, for different topologies of the incomplete networks and different PSMT protocols the adversary structures are quite different. Adversary structures for which we cannot realize secure function evaluation are called infeasible.

**The intermediate networks:** We replace one virtual edge at a time, from the original constructed network $N_C$, by simulation of perfect secure message transmission protocol on the incomplete network. We prove that the resulting intermediate protocol for the intermediate network is secure. Our proof uses the reduction argument. For this purpose, we define the functionality of the virtual edges so that it provides the adversary with more flexibility, compared to the case when the virtual edge is substituted by execution of PSMT protocol on the incomplete network. For example, an adversary for the former case (of virtual edges) gets to see the message being sent on the virtual edge earlier, then it would in the latter case. Thus, we start by handling strictly more powerful adversaries while considering greater number of virtual edges and consider progressively weaker adversaries as we replace the virtual edges one at a time. This is useful in reducing the complexity of the reduction argument.

**The necessity for super-round and slotting the super-round:** The original (unconditional) multiparty protocol, [BGW88], is synchronous and proceeds in rounds. We adapt this protocol to our case of incomplete networks, where many pairs of nodes (roughly $\Theta(n^2)$) cannot communicate with each other in a single round. Thus, we expand each round of the original protocol to a super-round, so that distant parties can communicate with each other by executing PSMT protocol on the incomplete network. When replacing virtual edges of the complete network (as described above) by simulation of PSMT on the incomplete network, we encounter the following technical difficulty:

When transmissions take place on two virtual edges simultaneously, then the messages sent/received/heard on one may influence the corruption of messages on another edge and vice versa. This is analogous to the phenomenon of "cross-talk" between close by wires caused by capacitance issues in VLSI chips. Handling this issue, for the case of active corruptions makes the proof of security of the underlying protocol complex. We go around this problem at the expense of increase in the round complexity of the protocol. We partition the super-round into slots so that each pair of parties has a dedicated non-overlapping slot. The time slots are wide enough to send messages on real edges or virtual edges. If there is a real edge in the underlying network for a given pair of parties then the length of slot is just one round. For the case of virtual edge the slot allotted is wide enough to execute the PSMT protocol on the underlying network.

**Why we need a more complex argument instead of a straightforward hybrid argument?** For malicious/active corruptions a straightforward hybrid argument to realize even Input indistinguishability type definition of privacy (as presented in [GO08]) encounters the following difficulty: During the execution of (unconditional) secure multiparty protocol, parties need to commit to certain secrets (e.g., the input value

during the input commitment phase of [BGW88]). This is achieved by executing a protocol for Verifiable secret sharing: The committing party shares a secret by sending evaluations of a polynomial to the rest of the parties. The parties then exchange sub-shares of these shares between themselves to make sure that the value committed to is valid and can be recovered by the rest of the parties. A malicious party may send incongruent shares of its secret to other honest parties (or potentially other corrupted parties). When these honest parties later tally the values of their sub-shares, they do not concur. The parties may then raise an alarm and incongruencies are reconciled by executing a "disavowal" protocol in which the committing party may have to reveal the correct shares of the contending parties to the rest of the parties. This depends on the choices made by the corrupted parties (as the honest parties always execute the protocol meant for them). Alternatively, a false alarm may be raised by a malicious party which can claim that its sub-share value does not match with that of another honest party (or perhaps another corrupted party). This may also result in execution of a "disavowal" protocol to reconcile the differences. The execution of the "disavowal" sub-protocol results in a different and lengthier sequence of steps, compared to when it is not executed at all. Thus, the actual size of the views of the parties, including those of the corrupted parties, depends on the number of times the disavowal protocol is executed. Also, the contents of the views of the parties depends on the particular sequence of steps of disavowal protocol executed. A straightforward hybrid argument cannot be used to argue indistinguishability between distributions of views, with such divergence.

## 3 Model and Definitions

In this section we review some standard notations and definitions for (unconditional) secure multiparty computation and $\mathcal{T}$-secure computation proposed in [Vay08].

**Definition 1.** *A function $\delta : N \rightarrow [0,1]$ is called negligible if for all $c > 0$ and for all large enough $k \in N$, we have $\delta(k) < k^{-c}$.*

**Definition 2.** *A distribution ensemble $X = \{X(k,a)\}_{k \in N, a \in \{0,1\}^*}$ is an infinite set of probability distributions, where $X(k,a)$ is associated with each $k \in N$ and $a \in \{0,1\}^*$. A distribution ensemble is called binary if it consists only of distributions over $\{0,1\}$.*

**Definition 3. (Statistical indistinguishability)** *Distribution ensembles $X$ and $Y$ are called statistically indistinguishable if for all sufficiently large $k$ and $a$, $SD(X,Y) = \frac{1}{2}\sum_a |Prob(X=a) - Prob(Y=b)| < \delta(k)$ for $\delta(.)$ a negligible function.*

$Power - Set(F)$ refers to the set of all subsets of a set $F$. A mixed network $N$ is referred to as a triplet $(V,E,E_d)$, where $V$ refers to a set of vertices, and $E$ refers to the set of undirected edges and $E_d$ refers to the set of directed edges.

### 3.1 Characteristics of the the adversary

If $\mathcal{A}$ corrupts a party *actively*, it gains complete control over the party its input value, its random tape, its program and is free to send arbitrary messages on the behalf of the party, while also receiving all the messages sent to the party by other parties. The party is said to be *passively* corrupted when the adversary just gains the privilege to receive all the inputs, outputs and messages exchanged by it with other parties.

We assume that the adversary is computationally unbounded. We consider corruptions of parties as well as channels, passively as well as actively. Towards this end we set up an *adversary structure*.

**Definition 4.** *Let $\mathcal{T} \subset \{(\mathcal{X}_p, \mathcal{X}_a, \mathcal{Y}_p, \mathcal{Y}_a) | \mathcal{X}_p, \mathcal{X}_a \subset V, \mathcal{Y}_p, \mathcal{Y}_a \subset V * V$, and $\mathcal{X}_p \bigcap \mathcal{X}_a = \phi, \mathcal{Y}_p \bigcap \mathcal{Y}_a = \phi\}$, to denote a set of quadruplets of subset of parties corrupted passively, subset of parties actively corrupted, subset of channels passively corrupted and a subset of channels actively corrupted. If the quadruplet of subset of parties corrupted passively, actively and subset of channels corrupted passively and actively by the adversary belong to $\mathcal{T}$, then the adversary is called $\mathcal{T}$-restricted.*

$\overrightarrow{C}$ is used to refer to a quadruplet of corruptions that belong to adversary structure $\mathcal{T}$. $\overrightarrow{C}$ is called *feasible corruption*, if there exists a subset of honest parties $H : |H| \geq \lfloor \frac{2n}{3} \rfloor + 1$, such that every two parties that belong to $H$ are connected via an uncorrupted channel. A *Feasible adversary structure* $\mathcal{T}$ is defined along the same lines.

### 3.2 Semi-formal definitions for $\mathcal{T}$-secure computation

We present semi-formal definitions (Please consult Section A for formal definitions) for $\mathcal{T}$-secure computation. The definitions are for stand alone model (taken from [Vay08]). However, it does not follow the traditional approach of defining an "Ideal process" and comparing it with a "Real process". The definition of privacy is an extension of the understanding on which Zero-Knowledge property of protocols are based (Please refer to Appendix Section C for an elaborate exposition of the definition).

**Definition for (unconditional) secure multiparty computation in the stand alone model** We consider two phase protocols $\Pi = (\Pi^1, \Pi^2)$: In the first phase parties commit to their input values and in the second phase parties evaluate function $f$ on the vector of input values committed to in the first phase.

We present a semi-formal definition for the canonical requirements of correctness and privacy of a two phase multiparty protocol, [GMW87]. The correctness is defined with respect to the vector of input values committed to in the input commitment phase of the protocol, where as privacy is defined for the entire protocol. First, we define correctness of the input commitment phase:

Semi-formally,

**Definition 5.** *$\Pi^1$ denotes the input commitment phase of a two phase protocol $\Pi = (\Pi^1, \Pi^2)$ if one can associate with it an n-variate function $reveal_{\Pi^1}()$, which when applied to the transcripts of the parties (with the condition that transcripts substituted for the corrupted parties are not necessarily genuine) generated by the execution of $\Pi^1$ reveals the vector of input values committed to by the parties. Furthermore, all the honest parties are able to commit to their actual input values (Corresponding formal Definition 11 appears in the Appendix).*

Next, we define two phase computation.

**Definition 6.** *Protocol $\Pi = (\Pi^1, \Pi^2)$ is a two phase multiparty protocol, where $\Pi^1$ is as qualified in Definition 5, if it satisfies the two conditions: (1) Correctness: Let $\overrightarrow{Inp} \leftarrow reveal(\overrightarrow{Trans})$ be the vector inputs committed to by $\Pi^1$. Then, $f(\overrightarrow{Inp}, \overrightarrow{r_1}) \approx \Pi^2(\overrightarrow{Inp}, \overrightarrow{r_2})$, for uniformly chosen sequences of random bits $r_1, r_2$ (2) Privacy: There exists a simulator which is given the input values committed to by the the corrupted parties, output value $f(\overrightarrow{Inp})$, which generates a distribution of views of the adversary that is indistinguishable from the distribution generated from real execution of the protocol $\Pi$ (Corresponding formal Definition 12 appears in the Appendix).*

**Definitions for $\mathcal{T}$-secure computation on incomplete networks** We extend the previous definition of the two-phase multiparty computation to incorporate the notion of sacrificed honest parties. Definition for correctness of input commitment phase incorporating the notion of sacrificed honest parties is as follows.

**Definition 7.** *Let $\Pi^1$ denote the input commitment phase of a two phase protocol $\Pi = (\Pi^1, \Pi^2)$ qualified by n-variate function $reveal_{\Pi^1}()$ as defined above in Definition 5. Furthermore (1) A subset H of honest parties of size at least $\lfloor \frac{2*n}{3} \rfloor + 1$ are able to commit to their actual input values and the value of $reveal_{\Pi^1}()$ is same irrespective of whether or not genuine transcripts are substituted for the corrupted and sacrificed parties. (2) $\Pi^1$ is allowed to fail with negligible probability (Corresponding formal Definition 13 appears in the Appendix).*

Now we define $\mathcal{T}$-secure computation incorporating the notion of sacrificed honest parties.

**Definition 8.** *Let $\Pi = (\Pi^1, \Pi^2)$ be a two phase multiparty protocol, Definition 7, executed on an incomplete network $N$, with $\mathcal{A}$ restricted to $\mathcal{T}$. The privacy and correctness condition for $\Pi$ are the same as for Definition 6 except that (1) Only statistical guarantees are made for the correctness condition (2) When proving the privacy condition only the input values committed by the sacrificed and corrupted parties (which are together less then or equal to $\lfloor \frac{n}{3} \rfloor$ in number) are given to the simulator (Corresponding formal Definition 14 appears in the Appendix).*

**An alternate view of the definition of privacy** We discuss how the definition of privacy in this work imply the input indistinguishability type definitions of privacy. This implies that the results in this work imply the realization of input indistinguishability type definitions of privacy for maliciously corrupted parties.

Our definition of privacy is that the distribution of the views of the adversary generated from a real execution of the multiparty protocol when the corrupted and sacrificed parties start with an initial vector of input values $\overrightarrow{y}_{P-H}$, commit to vector of input values $\overrightarrow{x}_{P-H}$ and receive output value $o$ is indistinguishable from a distribution of views that could be generated by a simulator *Sim* which is given these values only. This is interpreted to mean that the adversary can learn nothing more about the input values of the (unsacrificed) honest parties then what could be learned from the vector of input values $\overrightarrow{y}_{P-H}$, committed input values $\overrightarrow{x}_{P-H}$ and output value $o$ only ( This interpretation is along the lines of Zero-Knowledge Proofs).

Observe that for any two executions of the multiparty protocol for which the corrupted and sacrificed parties start with the same initial vector of input values $\overrightarrow{y}_{P-H}$, commit to the same vector of input values $\overrightarrow{x}_{P-H}$ and generate the same output value $o$, the views of the adversary are indistinguishable because in each case the views can be generated by the same simulator which is given these values only. In other words, the adversary cannot distinguish between two vectors of input values of the (unsacrificed) honest parties for which the vector of initial input values, vector of committed input values and the output value $o$ (which is same for all parties) of the remaining i.e., sacrificed honest parties and corrupted parties is same. This is input indistinguishability.

## 4 Setting up the communication infrastructure on incomplete networks

We describe a $\mathcal{T}_{\beta_{N,c}}$-Communicability Property of a network $N = (V,E)$. It essentially says that for every choice of quadruplet of corruptions $\overrightarrow{C} \in \mathcal{T}$, there exists a large enough subset of honest parties that can communicate securely with each other either by way of an edge of the incomplete network $N$ which corresponds to a secure channel, or by establishing a secure channel by utilizing the infrastructure of the incomplete network $N$. Note that for different choices of quadruplets $\overrightarrow{C} \in \mathcal{T}$ corrupted, the subset of honest parties $H$ for which we will be able to guarantee mutually secure communication can and will vary. So also $\mathcal{T}_{\beta_{N,c}}$-Communicability Property is going to hold for different subsets $H$. Our interest will only be in the statistical guarantees on the size of subset $H$ for which we can ensure this.

We just present semi-formal definitions of authentic and secure channels: Protocol $\beta_{N,p_u,p_v,c}$ establishes a $\mathcal{T}$-authentic channel between nodes $p_u$ and $p_v$ iff for all quadruplets $\overrightarrow{C} \in \mathcal{T}$ corrupted by $\mathcal{A}$ protocol $\beta_{N,p_u,p_v,c}$ establishes an authentic channel between $p_u$ and $p_v$. $\mathcal{T}$-secure channels are defined along the same lines (Please consult Appendix Section B for formal definitions).

Property $\mathcal{T}_{p_u,p_v,\beta,c}$ states the requirements for an incomplete network $N$, so that for all quadruplets $\overrightarrow{C} \in \mathcal{T}$ corrupted by adversary $\mathcal{A}$ secure channels can be established between $p_u$ and $p_v$ using the infrastructure of the incomplete network.

**Definition 9.** *Network N possesses Property $\mathcal{T}_{p_u,p_v,\beta,c}$ if the following two conditions hold true:*

1. *For every $\overrightarrow{C} \in \mathcal{T}$ corrupted by $\mathcal{A}$, there exists at least two node-disjoint paths with end points $p_u$ and $p_v$ such that the paths (a) are of length at most $c * \lg_2 n$, (b) consist only of honest nodes and secure channels.*
2. *There exists a (polynomial round) multiparty protocol $\beta$ that realizes $\mathcal{T}$-authentic channel from $p_u$ to $p_v$ and $p_v$ to $p_u$.*

Using the local property $\mathcal{T}_{p_u,p_v,\beta,c}$ of network $N$, which captures sufficient conditions for feasibility of establishing channels between two nodes $p_u$ and $p_v$ using the infrastructure of the incomplete network $N$, we define global properties of the incomplete network $N$, which makes statistical guarantees about the communication infrastructure of the entire network:

**Definition 10.** *Let $\mathcal{A}$ be a $\mathcal{T}$-restricted adversary. An incomplete network $N = (V,E)$ has a $\mathcal{T}_{\beta_N,c}$-Communicability Property iff*

*$\forall \overrightarrow{C} \in \mathcal{T}$ : There exists a large subset of honest parties $H : |H| \geq \lfloor \frac{2*n}{3} \rfloor + 1$ such that for every $\forall p_u, p_v \in V$, there exists a real edge $(p_u, p_v)$ in the network N or the network N possesses $\{\overrightarrow{C}\}_{p_u,p_v,\beta_N,c}$ Property 9.*

## 5 $\mathcal{T}$-secure multiparty computation on incomplete networks

This section presents the main theorem which stitches together other subsidiary claims. While we are able to present a very brief overview of the proof of the main theorem, the proofs of other subsidiary claims have been relegated to the Appendix. Subsection 4 is devoted to the subsidiary claims while Subsection 5 presents the main theorem in this work.

Let $\mathcal{A}$ be an adversary restricted to adversary structure $\mathcal{T}$.

Now we present the main theorem. It says that if network $N$ possesses $\mathcal{T}_{\beta_N,c}$-Communicability Property, then there exists a secure multiparty computation protocol $\Pi_N$ that $\mathcal{T}$-securely evaluates multivariate function $f$ on network $N$:

**Theorem 1.** *If network N possesses $\mathcal{T}_{\beta_N,c}$-Communicability Property 10, then there exists a two-phase multiparty computation protocol $\Pi_N$, that $\mathcal{T}$-securely evaluates function $f$, on network $N$, as according to Definition 14.*

The full proof of the theorem is relegated to the Appendix, Section F. We present a brief overview.

*Proof.* (Sketch) At a very high level, the proof of the main theorem proceeds as follows: We first invoke Theorem 8 for complete network $N_C$. Then, we inductively replace each non-existing edge of the incomplete network $N$ by a channel simulated by protocol $\gamma$ (this channel may behave as secure or authentic or even tamperable). We make sure that the correctness and privacy conditions hold true for the new multiparty protocol, which is to be executed over the new incomplete network, and for a new (lighter) adversary structure. More details follow.

The higher level construction of the protocol and the proof of its security proceed as follows:

Let $C_N = (V,E,E_{C_N})$ be the network with set of vertices $V$ constructed by connecting all the pairs of parties not already connected in the original incomplete network $N$, by *virtual* edges (Note, that since the virtual edges are directed, so for each undirected $(u,v) \in V * V - E$ two directed edges are added to $E_{C_N}$).

We fix an (arbitrary) order on all the (directed) virtual edges, in subset $E_{C_N}$, and proceed by induction as follows:

1. Base case: We start by constructing an adversary structure $\mathcal{T}_{C_N}$ for a complete network $C_N$ using the adversary structure $\mathcal{T}$ for the original incomplete network, network $N$, and $\mathcal{T}_{\beta_{N,c}}$-communicability property, 10. Then, we show that there exists a multiparty protocol $\Pi^0$, 14, that $\mathcal{T}_{C_N}$-securely evaluates $function f$ on the complete network $C_N$.
2. By induction hypothesis, we are given an intermediate network $N_i$ for which we are given a multiparty protocol $\Pi^i$ that $\mathcal{T}_{N_i}$-securely evaluates function $f$ on network $N_i$ (as according to Definition 14).
   We are required to present a construction of multiparty protocol $\Pi^{i+1}$ to be executed on network $N_{i+1}$ (where network $N_{i+1}$ is obtained by deleting the $i^{th}$ virtual edge from network $N_i$) and adversary structure $\mathcal{T}_{N_{i+1}}$ (obtained by removing edge $e_i$ from $\mathcal{T}_{N_i}$), such that protocol $\Pi^{i+1}$ $\mathcal{T}_{N_{i+1}}$-securely evaluates function $f$ on network $N_{i+1}$ (as according to Definition 14).
3. Lastly, note that inductive hypothesis is applied till there are no virtual edges left in the network, i.e., $N_j = N$ and $\mathcal{T}_{N_j} = \mathcal{T}$, the original adversary structure. The multiparty protocol $\Pi^N = \Pi^j$, $\mathcal{T}$-securely evaluate function $f$ on network $N$ (as according to Definition 14).

Loosely speaking, the induction hypothesis says that if we can realize $\mathcal{T}_{N_i}$-secure computation on incomplete network $N_i$ with $r$ real, $v$ virtual and $s$ simulated channel, then we can realize $\mathcal{T}_{N_{i+1}}$-secure computation on network $N_{i+1}$ with $r$ real, $v-1$ virtual and $s+1$ simulated edges. The proof of correctness and privacy of the intermediate protocols follow reduction arguments. We present an outline of the proof.

When considering $\mathcal{T}_{N_{i+1}}$-adversary $\mathcal{A}_{i+1}$ attacking protocol $\Pi^{i+1}$'s execution on network $N_{i+1}$, we need to create a $\mathcal{T}_{N_i}$-restricted adversary $\mathcal{A}_i$ which attacks protocol $\Pi^i$. Adversary $\mathcal{A}_i$ just internally simulates $\mathcal{A}_{i+1}$ and forwards messages generated by it to appropriate parties and vice versa. Now note that in network $N_i$ there is a virtual channel $e_i$ which is replaced by execution of PSMT protocol that achieves the same purpose as the virtual channel. Obviously, for the latter case the view generated of the honest parties as well as the adversary is different and grossly speaking larger then the former case. For this careful internal simulation is carried out depending on whether the virtual channel being replaced is secure, passively corrupted or actively corrupted, and whether the transmitting node is a corrupted party or an honest party. We show that the distribution of the views of $\mathcal{A}_{i+1}$ generated for these two cases is indistinguishable: (a) When $\mathcal{A}_{i+1}$ attacks the execution of the protocol $\Pi_{i+1}$ on network $N_{i+1}$ (b) When $\mathcal{A}_{i+1}$ is internally simulated by $A_i$ who attacks execution of protocol $\Pi_i$ on network $N_i$. For this purpose, it is inductively argued that the Vector of distribution of views of all the (sacrificed and unsacrificed) honest parties and $\mathcal{A}_{i+1}$ are indistinguishable for Case (a) and Case (b), after every "super-round". This constitutes the main body of the proof and it has the following highlightable features:

(1) Each super-round of protocol $\Pi^i$ (and protocol $\Pi^{i+1}$) is sliced into $n*(n-1)$ different slots. Thus, each ordered pair is allotted a dedicated slot for message transmission. The view of the adversary constructed up till $j^{th}$ super-round determines the messages the adversary chooses to send in the $j+1^{th}$ super-round. Inductively it is made sure that after each individual "slot" of the "super-round" also the distribution of the views of adversary $A_{i+1}$ is indistinguishable for the two Cases (a) and (b). (2) At the same time all the honest parties go through a clean up phase after every super-round. During the clean up phase all irrelevant intermediate messages generated in the process of execution of the PSMT protocol on the incomplete network are deleted.

Given the above condition it is relatively straightforward to prove that protocol $\Pi^{i+1}$ is secure. For this we need to show the correctness of commitment phase, correctness of computation phase and privacy of protocol $\Pi^{i+1}$:

1. To *prove the correctness of the Input commitment phase* we need to demonstrate a function $reveal_{\Pi^{i+1}}$ that satisfies the outlined characterization for the Input commitment phase. Recall that we show that the distribution of the views of the parties are indistinguishable for the two cases, after every super-round. This holds true after the Input commitment phase also. Thus the distribution of transcripts of the (unsacrificed) honest parties are also indistinguishable for the two cases (a) and (b), after the Input

commitment phase. Now recall that the transcripts of the unsacrificed honest parties are enough to reveal the set of input values committed to by all the parties. In particular, the same function $reveal_{\Pi^i}$ for protocol $\Pi^i$ will extract the same distribution of input values committed to by the parties. Thus, the function $reveal_{\Pi^{i+1}} = reveal_{\Pi^i}$ will satisfy the requisite characterization of the correctness of the Input commitment phase of protocol $\Pi^{i+1}$.

2. *Proof of correctness of the computation phase* follows from comparing the distribution of views of the honest parties for the two cases, at the termination of the protocol. The vector of distribution of views are indistinguishable for the two cases after every round. Conditioned to the fact that the inputs committed to in the Input commitment phase are identical, the outputs must also be the same for every deterministic function $f$ (and indistinguishable for the case of probabilistic function $f$). This completes the overview of the proof of correctness.
3. For the *Proof of privacy*, we transfer the work for the proof of correctness. By Induction Hypothesis, we have an appropriate simulator $S_i$ that simulated the view of $\mathcal{A}_i$'s attacking execution of protocol $\Pi^i$. Now realize that for the proof of correctness we show how $\mathcal{A}_i$ internally simulates adversary $\mathcal{A}_{i+1}$ and its distribution of views are indistinguishable for the two Cases (a) and (b). But this is sufficient for us to demonstrate a simulator $S_{i+1}$ that produces distribution of views of $\mathcal{A}_{i+1}$ attacking protocol $\Pi^{i+1}$'s execution on network $N_{i+1}$. Simulator $S_{i+1}$ is nothing but Simulator $S_i$ simulating $\mathcal{A}_i$, which in turn internally simulates $\mathcal{A}_{i+1}$ as described above. Proof of privacy for $\Pi^{i+1}$ follows from existence of simulator $S_i$ and the indistinguishability of the distribution of views of $\mathcal{A}_{i+1}$ for the two Cases (a) and Case (b).

□

## 6 $\mathcal{T}$-secure multiparty computation on a few specific classes of incomplete networks

In this section we describe a class of incomplete networks with which we associate appropriate adversary structures, for which the networks possess $\mathcal{T}_{\beta_{N,c}}$-Communicability Property. This sets in place the communication infrastructure necessary to execute multiparty protocols. We invoke the main theorem 1, to obtain multiparty protocols that $\mathcal{T}$-securely evaluate arbitrary $n$-variate function $f$ on the corresponding networks. For this we consider the (infinite) families of incomplete networks considered by [DPPU88], [Upf92] for the almost everywhere agreement problem.

1. In [Upf92], a protocol was presented that achieves $O(t)$-agreement, on a class of constant degree expander graphs, namely LPS expanders. The following Theorem 2 is a (minor) restatement of the theorem from [Upf92].

   **Theorem 2.** *There exists*
   *(a) Constant $\alpha > 0$, $\mu$, $q$ and $d$, independent of $t$ and $n$;*
   *(b) An $n$-vertex $d$-regular network $G$, which can be explicitly constructed;*
   *(c) A multiparty communication protocol $\beta(p_u, p_v, N)$ with round complexity $\alpha$.*
   *Such that for any subset $\tilde{C}$, $|\tilde{C}| \leq k * n$ for some constant $k$, of faulty nodes in $G$, communication protocol $\beta(p_u, p_v, N)$ there exists a subset of non-faulty nodes $H$, where $|H| \geq n - \mu * |\tilde{C}|$, such that $\beta(p_u, p_v, N)$ establishes $\{(\tilde{C}, \phi, \phi, \phi)\}$-authentic communication channel between $p_u$ and $p_v$. Furthermore, $\beta$ requires at most $q * \lg_2 n$ communication rounds.*

   The proof of Theorem 2 says that there exists at least one node disjoint path of honest nodes between every pair of parties $p_u$, $p_v$ from subset of honest nodes $H$, of length at most $d * \lg_2 n$. A more generic observation was made about Theorem 2, [Upf92] in [BBC+06]. Namely, if from an expander graph a (small) linear fraction of nodes are deleted, there still exists a large enough sized sub-network of non-faulty nodes which has good expansion properties. Since the good sub-network $G = (H, E')$ of our interest is an expander graph, it has several node disjoint paths passing through only non-faulty nodes of length $O(\lg_2 n)$, connecting every pair of nodes of $G$. From this we obtain the following:

(a) There exists at least two node disjoint paths of honest nodes between every pair of parties $u$, $v$ from subset $H$ of length at most $d * \lg_2 n$.
(b) There exists a multiparty protocol $\beta(p_u, p_v, N)$ that realizes $\{\tilde{C}\}$-authentic channel between parties $p_u$ and $p_v$ for $\forall p_u, p_v \in H$.

Let $\varepsilon = sup_{\{\chi | n - \mu * \chi * n \geq \lfloor \frac{2*n}{3} \rfloor + 1\}}$. Define the following adversary structure: $\mathcal{T} = \{(\tilde{C}, \phi, \phi, \phi) | \tilde{C} \subset V, |\tilde{C}| \leq \lfloor \varepsilon * n \rfloor\}$. We have the following corollary to Theorem 2:

**Corollary 1.** *Let N be a network from above Theorem 2. Then N possesses the $\mathcal{T}_{\beta,N}$-Communicability Property 10, for adversary structure $\mathcal{T}$.*

Putting together Corollary 1 with Theorem 1 we have:

**Theorem 3.** *There exists a multiparty protocol $\Pi$ to be executed on network N (LPS expander with n nodes, as in Theorem 2) which $\mathcal{T}$-securely evaluates function $f$, as according to Definition 14.*

2. Consider the recursive construction of committee networks of unbounded degree as defined in [DPPU88]. The description for one level of recursive construction is as follows: $V$ contains $n^{\varepsilon}$ committees, each committee further consists of $m = n^{\varepsilon}$ sub-committees (so on and so forth for $1/\varepsilon$ recursive steps). Inside, each committee all the sub-committees form a clique i.e., every two sub-committees $A_i$ and $A_j$ are connected. Each such connection represents a meta-edge between some matching of nodes of the sub-committees $A_i$ and $A_j$. For more details the reader is referred to [DPPU88].
Let $G_{\varepsilon}$ be a committee graph from [DPPU88] with $n$ nodes. The following theorem was proved in [DPPU88] for network $G_{\varepsilon}$:

**Theorem 4.** *For every $\varepsilon > 0$ there exists a constant $c = c(\varepsilon)$, graphs $G_{\varepsilon} = (V, E)$, $|V| = n$ of degree $O(n^{\varepsilon})$ and $t$-resilient $O(t)$-agreement protocol $t \leq c * n$.*

As a corollary to the proof of Theorem 4 from [DPPU88], we obtain the following:

**Corollary 2.** *There exists a constant $d$, and a family of multiparty protocols $\beta(p_u, p_v, N)$, to be executed on $G$, such that for every subset of nodes $\tilde{C} \subset V, |\tilde{C}| \leq c * n$ corrupted by $\mathcal{A}$, there exists a subset $S$ of honest nodes $S \subset V$ such that,*

*(a)* $|S| \geq n - O(t)$
*(b) There exists two paths of non-faulty nodes of length at most $d * \lg_2 n$ between $p_u, p_v$ for all $p_u, p_v \in S$.*
*(c) For all $p_u, p_v \in S$ multiparty protocol $\beta(p_u, p_v, N)$ realizes $\{\{tildeC, \phi, \phi, \phi\}\}$-authentic channel between $p_u$ and $p_v$.*

Define the following adversary structure: $\mathcal{T} = \{\overrightarrow{C} | \overrightarrow{C}[0] \subset V, |\overrightarrow{C}[0] \bigcup \overrightarrow{C}[1]| \leq \varepsilon * n, \overrightarrow{C}[1] = \overrightarrow{C}[2] = \overrightarrow{C}[3] = \phi\}$, for network $G$, which basically captures all subsets, of passive and active corruptions of parties, of size at most $\varepsilon * n$, such that $n - \mu * \varepsilon * n \geq \lfloor \frac{2*n}{3} \rfloor + 1$. We have the following corollary to the above Corollary 2:

**Corollary 3.** *Let $N_{\varepsilon}$ be the committee graph of n nodes and $\mathcal{T}$ be an adversary structure as defined above. Network N possesses $\mathcal{T}_{\beta,N}$-Communicability Property 10, for adversary structure $\mathcal{T}$.*

Putting together with Theorem 1, we obtain:

**Theorem 5.** *There exists a multiparty protocol* $\Pi$ *which* $\mathcal{T}$*-securely evaluates function* $f$ *on committee graph* $N_\epsilon$ *with n parties, for adversary structure* $\mathcal{T}$ *as defined above, as according to Definition 8.*

3. Results of similar nature, as described above, can be stated for butterfly graphs and constant degree expanders, which can tolerate $\frac{n}{\lg_2 n}$ and $n^\delta$ (for some constant $\delta$) corrupted nodes respectively, using the results in [DPPU88].

## References


[BBC+06] Amitabha Bagchi, Ankur Bhargava, Amitabh Chaudhary, David Eppstein, and Christian Scheideler. The effects of faults on network expansion. *Theory of Computing Systems*, 39:903–928, 2006.

[BG89] P. Berman and J. Garay. Asymptotically optimal distributed consensus. In *International Colloquium in Automaton, Languages and Programming, ICALP*, pages 80–94. Springer Verlag, 1989.

[BG90] P. Berman and J. Garay. Fast consensus on networks of bounded degree. In *International Workshop on Distributed Algorithms*, pages 321–333, 1990.

[BGW88] M. BenOr, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of Twentieth annual Symposium of Theory of Computation, STOC*, Chicago, Illinois, May 1988. Association for Computing Machinery.

[CCD88] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proceedings 20th Annual Symposium on Theory of Computing, STOC*, Chicago, Illinois, May 1988. Association for Computing Machinery.

[CFGN96] R. Canetti, U Fiege, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Twenty-eighth annual ACM Symposium on Theory of computing, STOC*, pages 639–648, 1996.

[DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Young. Perfect secure message transmission. *Journal of ACM, JACM*, 1993.

[Dol83] D. Dolev. The byzantine generals revisited. *Journal of Algorithms*, 1(3), 1983.

[DPPU88] C. Dwork, D. Peleg, N. Pippinger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM Journal on Computing*, 1988.

[FHM99] M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In *Advances in Cryptology, ASIACRYPT'99*, Lecture Notes in Computer Science, Singapore, November 1999.

[GM82] S. Goldwasser and S. Micali. Semantic security. In *Foundations of Computer Science, FOCS*. IEEE, 1982.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Symposium on Theory of Computing, STOC*. IEEE, 1985.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of nineteenth annual Symposium of Theory of Computation, STOC*. ACM, May 1987.

[GO08] J. Garay and R. Ostrovsky. Almost everywhere secure computation. In *Advances in Cryptology, EUROCRYPT*, 2008.

[GP96] O. Goldreich and E. Petrank. Quantifying knowledge complexity. In *Foundations of Computer Science, FOCS*. IEEE, 1996.

[HM97] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In *ACM Proceedings of Sixteenth Annual Symposium in principles of Distributed Computing*, 1997.

[HM00] M. Hirt and U. Maurer. Player simulation and general adversary. *Structures in Perfect Multiparty Computation, Journal of Cryptology*, 2000.

[HMP00] M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In *Advances in Cryptology, ASIACRYPT*, Lecture Notes in Computer Science, 2000.

[KKMO94] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *Foundations of Computer Science, FOCS*, 1994.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. In *ACM Trans on Programming Language and Systems*, volume 4, 1982.

[OR96] M. Ben Or and D. Ron. Agreement in the presence of faults, on networks of bounded degree. *Information Processing Letters, IPL*, 57(6):329–334, March 1996.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocol with honest majority. In *ACM Symposium in Theory of Computing, STOC*, 1989.

[Upf92] E. Upfal. Tolerating linear number of faults in networks of bounded degree. In *Symposium on Principles of Distributed Computing, PODC*, Vancouver, British Columbia, Canada, 1992. ACM.

[Vay06] Shailesh Vaya. *Almost everywhere secure computation*. PhD thesis, University of California, Los Angeles, December 2006.

[Vay08] Shailesh Vaya. Secure multiparty computation with man-in-the-middle attacks. In *manuscript*. available upon request, 2008.

[Yao82] A. Yao. Protocols for secure computation. In *Proceedings of twenty third annual Symposium on Foundations of Computer Science, FOCS*. IEEE, 1982.

# A Formal definitions for $\mathcal{T}$-secure multiparty computation

This section is from [Vay08], where definitions for $\mathcal{T}$-secure computation were presented for secure multiparty computation handling man-in-the-middle type attacks. We use the same definitional framework in this work also. We start by reviewing some standard terminologies used in secure multiparty computation.

## A.1 Some standard terminologies relevant to secure multiparty computation

Let $\Pi$ be a multiparty protocol executed by $\mathcal{P}$. We define the *View* of a player as the set of inputs, random bits used by the player and all the messages received by the player during the execution of the protocol. Likewise, the *View* of the adversary is the vector of views of the players, corrupted by it. Further, the distribution of the views of the players/adversary is defined as the distribution of these views generated from executing the multiparty protocol taken over the different random choices made by the players and the adversary. This distribution is defined for a vector of inputs given to the parties. Formally,

Let multiparty computation protocol $\Pi$ be executed by a set of players $\mathcal{P}$. $View_{p_j}^{\Pi,P,\mathcal{A}}(\overrightarrow{C}, \overrightarrow{I})$ refers to the random variable denoting the view of $p_j$, when multiparty protocol $\Pi$ is executed by the set of players $P$ with input vector $\overrightarrow{I}$, when adversary $\mathcal{A}$ corrupts quadruplet $\overrightarrow{C}$. Correspondingly, the random variable $\overrightarrow{View}_X^{\Pi,P,\mathcal{A}}(\overrightarrow{C}, \overrightarrow{I})$ denotes the vector of views of subset of players $X$, constituted from executing protocol $\Pi$ amongst set of parties $P$ with input vector $\overrightarrow{I}$. Along the same lines, we define distributions over these random variables as $\mathbf{View}_{p_j}^{\Pi,P,\mathcal{A}}(\overrightarrow{C}, \overrightarrow{I})$ and $\overrightarrow{\mathbf{View}}_{\overrightarrow{C}}^{\Pi,P,\mathcal{A},\overrightarrow{C}}(\overrightarrow{C}, \overrightarrow{I})$.

The traditional approach to define security of multiparty computation protocols is to describe a Real and Ideal processes and argue the inability of the adversary to distinguish between the two. The definitions in this work deviate from this methodology in a subtle way. We base our definitions on the following underlying understanding: Suppose a multiparty protocol is executed with some subset of corrupted parties. A view of the honest parties and corrupted parties is generated in the process. Based on this view, the adversary may try to extract knowledge about the inputs of other honest parties.

We require that an indistinguishable distribution of views of the adversary be computable just from the initial input values, committed input values and the output value of the corrupted parties. This amounts to arguing that the adversary can derive no more knowledge about the input values of the honest parties, then what can be computationally derived from these values alone. This concludes the proof of privacy property of the protocol. Inherently, the same argument underlies the traditional definitions of secure multiparty computation, [GMW87]: There it is shown that the view of the adversary is generated by the simulator using the different values and it is concluded that adversary can derive no more knowledge about the input values of the honest parties from its view, then what it could derive (computationally) from these initial input values and the output values of the corrupted parties.

As discussed before in our model some honest parties may be sacrificed and not required to receive the correct output values or maintain the privacy of their initial inputs. By demonstrating that an indistinguishable distribution of views of the adversary is computable from the initial input values, committed input values and the common output value of the sacrificed and corrupted parties, we conclude that the adversary can derive no more knowledge about the input values of the unsacrificed honest parties then what can be computationally derived from these values alone. The initial values and committed values of the sacrificed parties are considered sacrificed in this sense. This concludes the proof of the privacy property of the protocol.

We now present the definitions. We start by introducing minimal notations that will be used during the rest of the section. Let $\Pi = (\Pi^1, \Pi^2)$ refer to a two phase multiparty protocol. Let $\overrightarrow{y} = (y_1, y_2, \ldots, y_n)$ denote the vector of input values given to the parties $\mathcal{P}$, where $y_i = \overrightarrow{y}[i] \in \{0,1\}^*$ refers to the input value of the $i^{th}$ party $P_i$.

### A.2 Definition for (unconditional) secure multiparty computation for the stand alone model

We first present the definitions for the stand alone model.
Characterization of correctness of the input commitment phase of a multiparty computation protocol To characterize the correctness of the Input Commitment phase of a multiparty protocol the question to ask is how should the transcripts of the parties relate with each each other at the termination of the Input Commitment Phase. Note that only the transcripts of the honest parties should be relied upon to achieve this characterization because the transcripts of corrupted parties cannot be trusted and in fact can be manipulated arbitrarily by the adversary. Thus our characterization should allow the flexibility that the corrupted parties may possess arbitrary transcript values. The input commitment phase of the multiparty computation protocol should have the following two properties:

1. Honest parties are able to commit to their initial inputs: All the honest parties are able to successfully commit to their initial input values they start with, irrespective of the behavior of the adversary $\mathcal{A}$.
2. Binding for all parties: After the termination of the Input Commitment phase none of the parties are able to modify the input values committed to, irrespective of how the corrupted parties may behave from here on.

We formalize the above requirements for the correctness of the Input Commitment phase as follows: We associate an $n$-variate function $reveal_{\Pi^1}(.,.,\ldots,.)$ with $\Pi^1$, which when applied to the transcripts of the parties generated by execution of $\Pi^1$ reveals the vector of input values committed to by the parties. This function should have two additional properties (1) Committed values of the honest parties are the same as the initial input values they started with (2) The committed values of the corrupted parties, may be different from their initial input values, but are unmodifiable from after here on, irrespective of how the corrupted parties choose to behave. This is captured by requiring that the function $reveal_{\Pi^1}$ has the same output irrespective of what transcript values are substituted for the corrupted parties. Furthermore, as long as the true transcript values of at least $\lfloor \frac{2*n}{3} \rfloor + 1$ honest parties are specified as $\lfloor \frac{2*n}{3} \rfloor + 1$ inputs of $reveal()$, the order in which these inputs are specified and in which the transcript values of the corrupted parties is not important and does not affect the output value of the function $reveal_{\Pi^1}(.,.,\ldots,.)$. We are now ready to present a formal definition for the correctness of the input commitment phase.

Let $\Pi^1(\mathcal{P}, \overrightarrow{y}, \overrightarrow{r^1}, \mathcal{C}, \mathcal{A})$ refer to the vector of input values committed to by $\mathcal{P}$ on the execution of $\Pi^1$, starting with some vector of input values $\overrightarrow{y}$, randomness $\overrightarrow{r}$, when $\mathcal{A}$ corrupts subset $\mathcal{C} \subset \mathcal{P}$.

**Definition 11.** *Let $\Pi = (\Pi^1, \Pi^2)$ is a two phase multiparty protocol. The input commitment phase $\Pi^1$ is correct iff there exists an n-variate function $reveal_{\Pi^1} : \{\{0,1,\perp\}^*\}^n \rightarrow \{\{0,1\}^*\}^n$, which can be associated with $\Pi^1$ such that it satisfies the following properties.*

*Let $\overrightarrow{Trans} = \Pi^1_{Trans}(\mathcal{P}, \overrightarrow{y}, \overrightarrow{r^1}, \mathcal{C}, \mathcal{A})$, denote the vector of transcripts of $\mathcal{P}$, generated by the execution of $\Pi^1$. Then,*

1. *Let $\overrightarrow{x} = reveal_{\Pi^1}(\overrightarrow{Trans})$. $\forall P_i \notin \mathcal{C} : \overrightarrow{x}[i] = \overrightarrow{y}[i]$ i.e., honest parties are able to corrupt their initial input values.*
2. *$\forall \overrightarrow{Trans'} : [(\forall P_i \notin \mathcal{C} : \overrightarrow{Trans'}[i] = \overrightarrow{Trans}[i]) \rightarrow (reveal_{\Pi^1}(\overrightarrow{Trans'}) = reveal_{\Pi^1}(\overrightarrow{Trans}))]$ i.e., the transcripts of the honest parties are sufficient to extract the values committed to by all the parties irrespective of the transcripts of the corrupted parties.*

*Further, $\Pi^1(\mathcal{P}, \overrightarrow{y}, \overrightarrow{r^1}, \mathcal{C}, \mathcal{A}) = reveal_{\Pi^1}(\overrightarrow{Trans})$, is used to refer to the vector of input values committed by $\Pi^1$.*

*Remark 1.* A remark may be useful to view the essentials of function *reveal*(). Basically, *reveal*() is really only taking a "set" of transcripts as input. The ordering of the transcripts is not important for it to extract the actual "set" of input values committed to by the parties. But since we want an ordering of the output values which we can refer to, the input transcripts to reveal() are also ordered.

**Complete definition of vanilla MPC protocols for the stand alone model** Let $f : (\{0,1\}^*)^n \rightarrow \{0,1\}^*$ be an *n*-ary functionality. Let $\overrightarrow{I} = (i_1, i_2, \ldots, i_n)$ denote the vector of input values of the parties.

**Definition 12.** *Let $f$ be an n-ary function as defined above. Let $\Pi = (\Pi^1, \Pi^2)$ be a two phase multiparty protocol, as characterized in Definition 11. Then,* $\Pi$ **securely evaluates** *$f$ if the following conditions hold true $\forall C \subset P$ of parties corrupted by $\mathcal{A}$, for which $|P - C| \geq \lfloor \frac{2*n}{3} \rfloor + 1$:*

1. *Correctness: Let $\overrightarrow{x}$ refer to the vector of input values committed to by the parties on execution of $\Pi^1$.[1]. Then, for all honest parties $p_i \in P - C$ the following holds true:*

   $\Pi^2(\overrightarrow{x})_{p_i} = f(\overrightarrow{x})$
2. *Privacy: There exists a simulator Sim, which takes as input the subset $C$, $\overrightarrow{y}_C$, $\overrightarrow{x}_C$, $f(\overrightarrow{x})$, adversary program $\mathcal{A}$, and generates the view of the adversary $\mathcal{A}$, such that the distribution of the views of $\mathcal{A}$ generated from real execution of $\Pi$ is indistinguishable from the distribution of the views of $\mathcal{A}$ generated by Sim:*

   $S^{\mathcal{A}}(C, \overrightarrow{y}_C, \overrightarrow{x}_C, f(\overrightarrow{x})) \approx \overrightarrow{View}_C^{\Pi,\mathcal{A}}(C, \overrightarrow{y}, \overrightarrow{x}, f(\overrightarrow{x}))$

   *for all feasible adversaries $\mathcal{A}$*[2].

*Remark 2.* Note that the simulator is given both $\overrightarrow{y}_C$ and $\overrightarrow{x}_C$ i.e., the initial input values and the input values committed to by the corrupted parties and the output value $f(\overrightarrow{x})$. The simulator aborts and ignores those sessions when the corrupted parties commit to input values different then $\overrightarrow{x}_C$. For information theoretic regime there is no constraint on the running time of the simulator. The distribution is compared with the distribution of views of the adversary that are generated from the real execution of the protocol when the parties start with initial input values $\overrightarrow{y}$ and committed to $\overrightarrow{x}$. If the distributions generated from the two cases are proved indistinguishable, it amounts to saying that the adversary gains no more knowledge about the input values of other honest parties then what could be computationally derived just from the initial input values and the committed input values.

### A.3 Definition for secure multiparty computation on incomplete networks

To define security of multiparty computation protocols for incomplete network we first propose a model in which the adversary can corrupt a subset of channels as well. The channels may be corrupted passively or actively. If the channel between two parties is corrupted passively, then the new channel is behaviorally equivalent to an *authentic channel*. If the channel between two parties is corrupted actively, then the new channel is behaviorally equivalent to a *tamperable channel*. The ideal functionality of channels under different types of corruptions is discussed in Subsection B.1.

For this setting not all the honest parties may be guaranteed to receive the correct output values or be able to preserve the privacy of their input values. Honest parties, for which this is not ensured, are called

[1] As defined above, the vector of input values committed to by the parties is specified by vector $\overrightarrow{x} = \overrightarrow{\Pi^1(\mathcal{P}, \overrightarrow{y}, \overrightarrow{r^1}, C, \mathcal{A})} = reveal_{\Pi^1}(\overrightarrow{Trans})$

[2] The variable $\overrightarrow{View}_C^{\Pi,\mathcal{A}}(C, \overrightarrow{x}_C, \overrightarrow{y}_C)$ is as defined above, at the beginning of the section, and used rather canonically in cryptographic literature

*sacrificed*. Informally, a multiparty protocol $\Pi$ $\mathcal{T}$-securely evaluates function $f$ if for every feasible $\overrightarrow{C}$ corrupted by $\mathcal{A}$ there exists a subset of honest nodes $H, |H| \geq \lfloor \frac{2*n}{3} \rfloor + 1$, which achieve the canonical guarantees of multiparty computation. This is ensured if a subset of $\lfloor \frac{2n}{3} \rfloor + 1$ honest parties are able to communicate with each other via secure communication channels. The weakened guarantees of multiparty computation protocol are:

1. Correctness of the outputs of the sacrificed honest parties is not guaranteed.
2. Privacy of the inputs as well as outputs of the sacrificed honest parties is not required to be preserved. For this we require to demonstrate a simulator which is given the initial input values and committed input values of all the corrupted and sacrificed parties and the output value. The simulator should be able to simulate the view of adversary generated from real executions of the multiparty protocol when the initial input values and committed input values of the corrupted and sacrificed parties are the same. The understanding behind this definition of privacy has been discussed at the beginning of this section.

**Correctness of the input commitment phase:** The input commitment phase should have the following properties to be correct (1) All "unsacrificed" honest parties should be able to commit to their initial input values, irrespective of the behavior of the corrupted parties (2) Binding: The honest parties share the committed input values of all the parties, which cannot be modified from after here on irrespective of the behavior of the malicious parties (3) The phase may fail with some negligible probability.

Let $\mathcal{A}$ be an adversary restricted to adversary structure $\mathcal{T}$, Definition 4. Let $\mathcal{P}$ be a set of parties. $N$ refers to the network, $\mathcal{P}$ refers to the set of participating parties. $\overrightarrow{C}$ refers to the quadruplets of parties and channels corrupted by the adversary $\mathcal{A}$.

**Definition 13.** *Let $\Pi = (\Pi^1, \Pi^2)$ be any two phase multiparty protocol. The input commitment phase $\Pi^1$ is correct if there exists an n-variate function $reveal_{\Pi^1} : \{\{0,1,\perp\}^*\}^n \rightarrow \{\{0,1\}^*\}^n$, such that the following conditions hold true for all feasible $\overrightarrow{C}$ corrupted by $\mathcal{A}$.*

*Let $\overrightarrow{Trans} = \Pi^1_{Trans}(\mathcal{P}, \overrightarrow{y}, \overrightarrow{r^1}, \overrightarrow{C}, \mathcal{A})$, denote the vector of transcripts of parties $\mathcal{P}$, generated by the execution of $\Pi^1$. Let $\overrightarrow{x} = reveal_{\Pi^1}(\overrightarrow{Trans})$. There exists a subset of (unsacrificed) honest parties $\mathcal{H} \subset \mathcal{P}$, $|\mathcal{H}| \geq \lfloor \frac{2*n}{3} \rfloor + 1$ such that:*

1. *Honest parties are able to commit to their initial inputs: $\forall P_i \in \mathcal{H} : x_i = y_i$, with probability greater than $1 - \mu(n)$, for some negligible function $\mu(.)$.*
2. *Binding for all the parties: $\forall \overrightarrow{Trans'} : (\forall P_i \in \mathcal{H} : \overrightarrow{Trans'}[i] = \overrightarrow{Trans}[i]) \rightarrow (reveal_{\Pi^1}(\overrightarrow{Trans'}) = reveal_{\Pi^1}(\overrightarrow{Trans}))$.*

**Notation:** The vector of input values committed to i.e., $reveal_{\Pi^1}(\overrightarrow{Trans})$ is denote by $\Pi^1(\mathcal{P}, \overrightarrow{y}, \overrightarrow{r^1}, \overrightarrow{C}, \mathcal{A})$.

We allow the input commitment phase of protocol for incomplete networks to fail with a negligible probability. This is because the protocol uses (almost) secure channels to transmit messages, which may fail to deliver the correct messages with a negligible probability i.e., behave as corrupted channels with this negligible probability. Hence, the commitment phase may also fail with a negligible probability. We are ready to present a full formal definition for $\mathcal{T}$-secure multiparty computation:

Let $f : (\{0,1\}^*)^n \rightarrow \{0,1\}^*$ be an $n$-ary functionality.

**Definition 14.** *Let $f, \overrightarrow{y}, P, \mathcal{A}, \overrightarrow{C}$ be defined as above.*

*Let $\Pi = (\Pi^1, \Pi^2)$ be a two-phase multiparty computation protocol, 13. Protocol $\Pi$, $\mathcal{T}$-securely evaluates function $f$ if there exists a simulator Sim, such that for all $\overrightarrow{C} \in \mathcal{T}$, (Subsection 4, corrupted by $\mathcal{A}$, there exists a subset of parties, $H \subset P - \overrightarrow{C}[0] - \overrightarrow{C}[1]$, $|H| \geq \lfloor \frac{2n}{3} \rfloor + 1$ the following holds true:*

1. *Correctness: Let $\overrightarrow{x}$ be the vector of input values committed to by the parties, after the execution of input commitment phase $\Pi$[13]. Then, for all $p_i \in H$ :*
   $\Pi^2(\overrightarrow{x}, \overrightarrow{C}, \mathcal{A}, \overrightarrow{r^2})_{p_i} = f(\overrightarrow{x})$[4]
2. *Privacy: Simulator Sim, takes as input $\overrightarrow{C}$, $\overrightarrow{y}_{P-H}$, $\overrightarrow{x}_{P-H}$, output $f(\overrightarrow{x})$, adversary program $\mathcal{A}$ and generates a distribution of views of $\mathcal{A}$, such that:*

   $Sim^{\mathcal{A}}(\overrightarrow{C}, \mathcal{T}, N, \overrightarrow{y}_{P-H}, \overrightarrow{x}_{P-H}, f(\overrightarrow{x})) \approx \overrightarrow{View}^{\Pi,\mathcal{A}}_{\overrightarrow{C}}(\overrightarrow{C}, \overrightarrow{x}, \overrightarrow{y}, f(\overrightarrow{x}), \overrightarrow{r})$.

   *for all feasible adversary structures $\mathcal{T}$.*

*Remark 3.*

1. The simulator is required to produce a distribution of the views of the adversary $\mathcal{A}$. This distribution is to be shown indistinguishable from the distribution of the views of the adversary generated from real execution of the protocol given that the initial input values and the committed input values of the parties are $\overrightarrow{y}$ and $\overrightarrow{x}$ and output is $f(\overrightarrow{x})$. This is supposed to capture the property that the adversary does not gain any more knowledge about the input values of unsacrificed honest parties, then what can be computationally derived from just $\overrightarrow{y}_{P-H}$ and $\overrightarrow{x}_{P-H}$ and $f(\overrightarrow{x})$.
2. The simulator is given the initial input values and the committed input values that the corrupt and sacrificed parties start i.e., $\overrightarrow{y}_{P-H}$ and $\overrightarrow{x}_{P-H}$ and the output value $f(\overrightarrow{x})$. We have already discussed the relevance of providing the simulator with these values in order to produce valid distribution of views of the adversary. The simulator starts the simulation by initializing the input values of the corrupt and sacrificed parties as above. At the end of the first phase the corrupt and sacrificed parties have committed to some input values. If they are the same as $\overrightarrow{x}_{P-H}$, then the simulator proceeds further, else outputs $\perp$ and aborts the current simulation. The simulator needs to only produce valid distributions conditioned to the fact that that simulation output is not $\perp$. For information theoretic regime there is no constraint on the running time of the simulator.

### A.4 Alternate view of the definition of privacy in this work

There is an alternate way to view the definitions of privacy in this work. For this let us review what we require for the privacy property to hold: There exists a simulator such that for all initial input values $\overrightarrow{y}$ and committed input values $\overrightarrow{x}$ for feasible corruptions the following holds true:
$Sim^{\mathcal{A}}(\overrightarrow{C}, \mathcal{T}, N, \overrightarrow{y}_{P-H}, \overrightarrow{x}_{P-H}, f(\overrightarrow{x})) \approx \overrightarrow{View}^{\Pi,\mathcal{A}}_{\overrightarrow{C}}(\overrightarrow{C}, \overrightarrow{x}, \overrightarrow{y}, f(\overrightarrow{x}), \overrightarrow{r})$.

Consider another set of vectors $\overrightarrow{x'}$ and $\overrightarrow{y'}$, such that $\overrightarrow{y'}_{P-H} = \overrightarrow{y}_{P-H}$ and $\overrightarrow{x'}_{P-H} = \overrightarrow{x}_{P-H}$ and $f(\overrightarrow{x}) = f(\overrightarrow{x'})$. For the privacy property to hold we have that:
$Sim^{\mathcal{A}}(\overrightarrow{C}, \mathcal{T}, N, \overrightarrow{y'}_{P-H}, \overrightarrow{x'}_{P-H}, f(\overrightarrow{x'})) \approx \overrightarrow{View}^{\Pi,\mathcal{A}}_{\overrightarrow{C}}(\overrightarrow{C}, \overrightarrow{x'}, \overrightarrow{y'}, f(\overrightarrow{x'}) = f(\overrightarrow{x}), \overrightarrow{r})$.

Now observe that the inputs of the simulator *Sim* for both the cases are same as $\overrightarrow{y}_{P-H} = \overrightarrow{y'}_{P-H}$, $\overrightarrow{x}_{P-H} = \overrightarrow{x'}_{P-H}$ and $f(\overrightarrow{x}) = f(\overrightarrow{x'})$, which implies that the distributions generated by the simulator for the two cases are also indistinguishable which in turn implies that:
$\overrightarrow{View}^{\Pi,\mathcal{A}}_{\overrightarrow{C}}(\overrightarrow{C}, \overrightarrow{x}, \overrightarrow{y}, f(\overrightarrow{x}), \overrightarrow{r}) \approx \overrightarrow{View}^{\Pi,\mathcal{A}}_{\overrightarrow{C}}(\overrightarrow{C}, \overrightarrow{x'}, \overrightarrow{y'}, f(\overrightarrow{x}), \overrightarrow{r})$.

Let us put other known facts about the relations between $\overrightarrow{x}, \overrightarrow{y}, f(\overrightarrow{x})$ and $\overrightarrow{x'}, \overrightarrow{y'}, f(\overrightarrow{x'})$ to interpret what is achieved here: As long as the initial input values, committed input values of the sacrificed parties and the corrupted parties, and the output values are same, the distribution of view of the adversary (which is just the vector of views of the corrupted parties) are indistinguishable for the two cases. What is essentially implied is input indistinguishability.

[3] $\overrightarrow{x} \leftarrow \Pi^1(\overrightarrow{y}, \overrightarrow{C}, \mathcal{A}, \overrightarrow{r^1})$, as defined above
[4] The equality condition can be relaxed for probabilistic function to $\approx$.

### A.5 Further refinement of the definition of security

We have noted previously that a 'sacrificed' party may just have the correctness property sacrificed or just the privacy property sacrificed or both the properties sacrificed. Our definitional framework can be easily extended to take into account such refinements with minor adaptations.

## B Ideal functionality for channels, $\mathcal{T}$-authentication channels and $\mathcal{T}$-secure channels

In this section we present the ideal functionality for virtual channels. Subsequently we also present definitions for $\mathcal{T}$-authenticated and $\mathcal{T}$-secure channels on incomplete networks.

### B.1 Modelling channels on incomplete networks

Channels realized by physical infrastructure like LAN, fibre optic cables or even WAN can be modelled as synchronous behavioral entities. We abstract out the functionality of a channel by describing an Ideal Functionality that captures the essentials of a channel's behavior. A directed channel $e_{u,v}$ from party $p_u$ to party $p_v$ if uncorrupted behaves as a secure channel. The message received by the channel $e_{u,v}$ from party $p_u$ is forwarded to party $p_v$ after a few rounds $r$. This number $r$ is hardwired in the channel. A channel may be passively or actively corrupted by the adversary. When passively corrupted the message sent on the channel is also revealed to the adversary by the channel. When the channel is actively corrupted the message sent on the channel may be corrupted by the adversary before it is finally sent by the channel to the receiving party.

All edges of the incomplete networks will be modelled as secure channels. If the edge of the network is undirected, then it will correspond to two channels, a forward directed channel and a backward directed channel both of which are secure channels.

Let $r$ be a constant, greater than 6, which will be specified later. Let $\mathcal{A}$ corrupt $\overrightarrow{C}$, as described above, 4. The directed channel from $S$ to $R$ is referred to by $F_d^r(S,R,edge_{id})$ and the ideal functionality for this directed channel from $S$ to $R$ is as follows:

**Definition 15.** *$F_d^r(S,R,edge_{id})$, denotes a channel from S to R, with unique identity $edge_{id}$, in the synchronous setting and executes as follows:*

1. *$edge_{id} \in \overrightarrow{C}[3]$: (Passive corruption) If message $(S,R,mesg-id,m)$ is received from party S in round i, then $F_d^r(S,R,edge_{id})$ records it and forwards the message $(S,R,mesg-id,m)$ to $\mathcal{A}$ in round $i+1$, and to party R in round $i+r-1$.*
2. *$edge_{id} \in \overrightarrow{C}[2]$: (Active corruption)*

   (a) *If $(S,R,mesg-id,m)$ is received from S in round i, $F_d^r(S,R,edge_{id})$ records the message, round number etc. and forwards the arrival note $(S,R,mesg-id,ARRIVAL)$ to $\mathcal{A}$ in round $i+1$.*
   (b) *In round $i+j$ for $j>1$ $\mathcal{A}$ sends the message $(mesg_id,READ-N-CORRUPT)$ or the message $(mesg_id,JUST-CORRUPT)$ to $F_d^r(S,R,edge_{id})$ depending on whether $\mathcal{A}$ intends to corrupt the message in plaintext or $\mathcal{A}$ intends to corrupt the message when hidden under a one time pad (where the latter case models partially corrupted channel). If $j<r-4$, then $F_d^r(S,R,edge_{id})$ records this message and proceeds further, else ABORT.*
   (c) *In round $i+j+1$, $F_d^r(S,R,edge_{id})$ gives message $(S,R,edge_{id},mesg)$ to $\mathcal{A}$, where $mesg = m \oplus onetimepad$ (for a uniformly chosen sequence of random bits onetimepad that is also recorded with the channel) if the channel $F_d^r(S,R,edge_{id})$ received the choice $READ-N-CORRUPT$ in the last round or $mesg = m$ if the channel received the choice $JUST-CORRUPT$ in the last round.*

(d) *In round k, $k \leq i+r-2$, $F_d^r(S,R,edge_{id})$ receives message $(S,R,mesg-id,m')$ to be sent to party R from $\mathcal{A}$. $F_d^r(S,R,edge_{id})$ checks validity, time stamp's etc of the message from previous records. If the choice of $\mathcal{A}$ was JUST−CORRUPT then $F_d^r(S,R,edge_{id})$ forwards message $(S,R,mesg_id,onetimepad \oplus m')$ to R, else it forwards $(S,R,mesg_id,m')$ to R in round $i+r-1$.*

3. *(Secure channel) If message $(S,R,mesg-id,m)$ is received from party S in round i, $F_d^r(S,R,edge_{id})$ records it and forwards the message $(S,R,mesg-id,\perp,|m|)$ to $\mathcal{A}$ in round $i+1$, and message $(S,R,mesg-id,m)$ to party R in round $i+r-1$.*

A few remarks are in place.

*Remark 4.*

1. The complication in the description of active corruption arises because the adversary in control of public discussion channel may not make all the non-faulty paths invalid. In such a case the adversary does not get to learn the message sent on the channel while is still able to mangle the message being sent.
2. In the proof of the main theorem, we use networks that have two different types of channels: *real* and *virtual*. Messages sent along either of the channels may take different number of rounds to reach the other party. The need for this distinction is made evident from our need of these channels in the proof of the main theorem in which we replace each virtual channel by simulation of PSMT protocol that uses the underlying incomplete network of real channels.
3. Note that when the edge $F_d^r(S,R,edge_{id})$ is passively or actively corrupted, the adversary $\mathcal{A}$ is given as much flexibility as possible to corrupt the message appropriately i.e., the adversary is given as many number of rounds to corrupt the message before forwarding the message to the receiver. This results in modelling stronger attack. This flexibility to the adversary is required as we start by assuming a network that has several virtual channels and a secure protocol for this network. As we replace the virtual channels with simulations of PSMT establishing channels employing the infrastructure of the incomplete network, the adversary is tied and becomes weaker. This is done so that we can use the claim of security for the former network and adapt it to prove security for the latter more realistic network but with more restricted adversary.

### B.2 Definition of $\mathcal{T}$-secure channel

Loosely speaking, a multiparty protocol realizes $\mathcal{T}$-secure channel between nodes *u* and *v*, if it realizes a secure channel between *u* and *v*, for all $\overrightarrow{C} \in \mathcal{T}$, corrupted by the adversary $\mathcal{A}$. There are no guarantees about the type of channel established when $\overrightarrow{C} \notin \mathcal{T}$.

There are two properties of a secure channel: *Correctness* and *Privacy*. Our interest is to realize almost correct channels which suffice for our purpose. Privacy property of channels is formalized along standard lines: A channel is called private iff there exists a (probabilistic) polynomial time simulator *Sim* which generates a distribution of views of adversary $\mathcal{A}$ which is indistinguishable from the distribution of views of $\mathcal{A}$ generated during real executions of the protocol. For technical reasons we demonstrate a simulator even when the channel realized is not private or authentic. For such cases the simulator may be given the message being transmitted on the channel. This simulator might be invoked to prove security of higher level protocols.

Let the adversary be restricted to adversary structure $\mathcal{T}$. Let $N=(V,E)$ be an incomplete network.

Let $\mathcal{T}$, $N=(V,E)$ be as above. Let $\overrightarrow{I}=(\perp,\perp,\ldots,m_u,\perp,\ldots,\perp)$ and $\overrightarrow{O}=(\perp,\perp,\perp,\ldots,m_v,\perp,\ldots,\perp)$ denote vector of inputs and outputs respectively, where $\overrightarrow{I}[i]$ and $\overrightarrow{O}[i]$ denote the input and output values of the $i^{th}$ party. Let $\overrightarrow{View}_C^{\gamma,P}(\overrightarrow{I})$ be defined as above. Let *l* be a security parameter. Let $\mu(.,.)$ be a negligible function.

**Definition 16.** *($\mathcal{T}$-secure channel) Multiparty protocol $\gamma(p_u,p_v,N,s,c)$ realizes $\mathcal{T}$-secure channel between nodes u and v on network N, if for all input vectors $\overrightarrow{I}$, quadruplet $\overrightarrow{C}$ corrupted by $\mathcal{A}$, such that $\overrightarrow{C} \in \mathcal{T}$, the*

*following conditions hold true:*

1. *Correctness: $I[u] = O[v]$, with probability $\geq 1 - \mu(l, |I[u]|)$ for all sufficiently large $l$.*
2. *Privacy: There exists a PPT simulator Sim that takes as input, the network topology $N$, $\overrightarrow{C}$, the program of adversary $\mathcal{A}$, size of input message $|I[u]|$ and runs in time polynomial in the running time of $\mathcal{A}$ and generates a distribution of views of $\mathcal{A}$ such that:*

   $Sim^{\mathcal{A}}(\overrightarrow{C}) \approx \overrightarrow{View}^{\Pi,P}_{\overrightarrow{C}}(\overrightarrow{I})$.

### B.3 Definition of $\mathcal{T}$-authentic channel

For authentic channels we just need to ascertain the correctness property. We add a redundant *simulatability* condition which says that the view of the adversary should be simulatable. There are two cases: (1) When the sender is corrupted the simulator does not need the message (2) When the sender is not corrupted the simulator is given the message to be sent to the receiver. Ultimately this simulator will be invoked by a simulator at a higher level, which will provide it with the requisite message to be sent.

**Definition 17.** *($\mathcal{T}$-authentic channel) Protocol $\gamma(p_u, p_v, N, s, c)$ realizes $\mathcal{T}$-authentic channel between $p_u$ and $p_v$ on network $N$, if for all input vectors $\overrightarrow{I}$, all quadruplets $\overrightarrow{C}$ corrupted by $\mathcal{A}$, such that $\overrightarrow{C} \in \mathcal{T}$, the following hold true:*

1. *Correctness: $\overrightarrow{I}[p_u] = \overrightarrow{O}[p_v]$.*
2. *Simulatability: There exists a simulator Sim that runs in time polynomial in the running time of $\mathcal{A}$ and generates a distribution of views of $\mathcal{A}$ that is indistinguishable from the distribution of views of $\mathcal{A}$ generated from real execution of $\gamma$. Furthermore, the simulator is given the input $inp \neq \perp$ iff $p_u$ is not corrupted. Namely,*

   $Sim^{\mathcal{A}, p_u, p_v, N}(\overrightarrow{C}, inp, l, c) \approx \overrightarrow{View}^{\Pi,\mathcal{P}}_{\overrightarrow{C}}(\overrightarrow{I})$.

## C Conceptual exposition of the definition of privacy in this work

We examine the role of simulator in several different definitions of security of two party or multiparty protocols and how demonstrating a simulator for these applications is supposed to be understood. We then present a meta-definition which captures the bare bone "skeletal" of these definitions. We then explain how the understanding subsumed in the meta-definition is employed to define security of these protocols and also this work.

The celebrated work on Zero Knowledge Proofs introduced the fundamental notion of knowledge complexity and defined 0-Knowledge Proofs (ZKP)[5]. Loosely, a protocol is called a 0-KP if the verifier does not gain anything from interacting with the prover which it could not have generated by itself, except for one bit i.e., validity of some statement. This is formalized as follows: A protocol is ZKP iff there exists a PPT simulator which given the inputs available to both the parties, the verifier's program and any auxiliary values with the verifier, can generate a distribution of transcripts of the verifier which is indistinguishable from the distribution of transcripts generated in real execution of the protocol. Let us examine this closely by considering an example.

[5] One may want to keep in mind the following sentence from page 295 (second last paragraph) of the historic paper [GMR85] while formulating the notion of knowledge, "With this in mind we would like to derive an upper bound (expressed in bits) for the amount of knowledge that a polynomially bounded machine can extract from a communication. Further review the definition for $L \in KC(f(n))$

Consider a ZKP in which the verifier is given an auxiliary input *Aux* but has limited space (i.e., space bounded). The (cheating) verifier tries to use the auxiliary input in order to extricate some extra knowledge while executing the protocol with the prover. As a result a transcript $T$ is generated with the verifier. Afterwards, due to space constraints the verifier deletes the auxiliary input *Aux*. Clearly, now the verifier does not seem to be able to generate an indistinguishable looking transcript of the verifier, because the verifier does not have the auxiliary input *Aux* with it, which the simulator takes as input. Does the same transcript $T$ which was earlier considered to convey 0-knowledge, now represent anything more to the verifier ? Why or why not? How much knowledge is contained in the transcript (of and) for the verifier?[6]

A statement that is consistently true both before and after the verifier deletes the auxiliary input *Aux* from its tape is: Whatever knowledge the verifier could (PPT) computationally derive from the transcript $T$, can also be computationally derived just from the initial input value which is given to both parties, the verifier's program and the auxiliary input value *Aux*. When the verifier also possessed the auxiliary input then this could be interpreted to mean that $T$ contained 0-knowledge for the verifier. Similar understanding is used to define the privacy property of multiparty protocols proved secure via simulation paradigm: An ideal process and real process are described. In the ideal process the power of the adversary looks "somewhat" curtailed and parties are given access to an ideal functionality, while the simulator simulates the protocol execution and generates views of all the parties. Distribution of these views of the adversary generated from the two processes are proved to be indistinguishable for the two cases to claim the privacy property of the protocol. But what is really implied in inferring the privacy property from such a proof, as in [GMW87] ? It is that the view of the adversary (for the ideal process case) could be generated by a (PPTM or just TM) simulator which is given access only to some input values and output values of the corrupted parties. Nevertheless, aside from these input and output values there are many other messages that are part of the view of the adversary even for the case of ideal process. So the underlying understanding being made is: The adversary cannot derive any more knowledge about the input values of the honest parties, then what can be (PPT) computationally derived just from its own input and output values. This is interpreted as the privacy property of the protocol.

Towards this end we intend to capture the essence of what demonstrating a simulator implies in definitions of ZKP and some other definitions of multiparty protocols (like the original definition in [GMW87]). The following meta-definition of simulator chaffs of unnecessary technical jargons, except for capturing the essential skeletal of these definitions of security:

*A simulator is hypothetical mental construct which is used to prove properties that should exist about the relations between input values, intermediate values and output values generated by the execution of a multiparty protocol with a given adversary.*

Let us see how we arrived at this meta-definition. First see that the distribution of transcripts generated by simulator *Sim* by interacting with verifier's program $V$ can be produced just by a single Turing Machine $U$ which is given the following auxiliary inputs: A string of bits encoding the program of the simulator *Sim*, a string of bits encoding the program of the verifier *Ver*. The Turing machine $U$ takes as input, the input value $I$ and the auxiliary value of the verifier *Aux*. $U$ is also given a random tape. $U$ simulates internally the interaction between the simulator and the verifier by interpreting the strings *Sim* and *Ver* as two procedures which have separate tapes/memories for performing read and write, but who are also given a common shared memory corresponding to the interactive tapes for communication. $U$ detects when the simulation has failed and verifier needs to be rewound to an earlier state and does this when necessary by maintaining a stack. Finally, $U$ outputs a distribution of transcripts. It is easily seen that $U$ runs in time polynomial in the running times of the original simulator *Sim* and the verifier *Ver* and produces the output in one shot without interactive computation of any kind.

[6] Going by the "hint" formulation of quantifying knowledge, [GP96], certainly the transcript contains no more knowledge then the size of the input value and the auxiliary input *Aux*. We mention this as a perspective of what's being asked or understood. That's all.

The output of Turing Machine $U$ on input $I, Aux$ is the requisite distribution $D_u^I$ i.e., $U(I,Aux)_r = D_u^I \approx P \leftrightarrow V(I,Aux)$. Looked another way $U$ is a Turing Machine which is just computing the value of a probabilistic function $func(.,.)$ whose domain is all possible tuplets that correspond to values that can be assigned to $I, Aux$ and range is a distribution of transcripts of the verifier. Furthermore, $func(I,Aux)_r \approx P \leftrightarrow V(I,Aux)$. The property of function $func(.,.)$ that it is PPT computable is interpreted to mean that the verifier does not derive anything more computationally from interacting with the prover, then what could be derived by the verifier itself. We emphasize the point here that the verifier may or may not have access to the simulator or the auxiliary values to participate in the simulation at the time of simulation. However, it is the property of the function func(.,.) i.e., PPT computability of $func(.,.)$ that matters here and which we care about. In other words, the simulator is merely an "abstract" construct which has nothing to do with what may or may not be achievable/available in reality and is used only to demonstrate how the initial input, auxiliary input, string encoding verifier's program relate to the intermediate and output values i.e., the transcripts generated in the execution of the protocol. What is achieved by demonstrating such a simulator is a proof of some properties of the "existing" relations between the different values possessed by possibly different parties. This methodology is "constructive" (we demonstrate an algorithm). The interesting property that the function $func(.,.)$ possesses from our perspective is that it is PPT computable.

This understanding is extended to the case of multiparty protocols in a straightforward manner. In multiparty protocols the adversary can be in control of several parties. Here too a simulator may be demonstrated that is given just the input values, intermediate values and output values to compute a distribution of (entire) views of the adversary. These values may or may not be available with the verifier. But what is established by the demonstration of such a simulator is that the output of the simulator given these values and the final output transcript of the verifier are related by way of computational indistinguishability. What is of importance here is the interpretation of proving such a fact. What does it amount to? Simulator may not exist in reality or it may not be feasible to execute the simulator in reality due to whatever reasons: may be because some values which we are interested in relating to belong to the honest parties (which may not cooperate) or may not be available with the verifier. However, these proofs are interpreted as properties of cryptographic protocols. For example, if a simulator is demonstrated which is given some input values, some intermediate values and output values (generated in real execution of the protocol) to generate a distribution of the views of the adversary, then this inference is made along the same lines as the example above i.e., The adversary cannot derive anything more computationally about the input values of honest parties, then what could be derived from these input values, intermediate values and output values alone.

This completes the discussion of the underlying understanding behind the definitions of security in this work. One may find it interesting to relate the discussion with measures of knowledge complexity that have been studied by Goldreich Petrank, [GO08]. One of the complexity measures proposed there was to give the simulator a hint. The knowledge complexity of a given "value" is defined as the length of the shortest hint required to for a PPTM to generate the appropriate value (or an indistinguishably looking distribution of them). This formulation is sufficient for one to define security of protocols the way it has been done here. However we emphasize, our interest is not in quantization of knowledge in terms of bits, yet be able to make such statements like, the adversary cannot computationally derive anything more from its view about the inputs of some other (honest) parties, then what it could just computationally derive from *x*, *y* or *z*, which as pointed above is the underlying understanding behind the simulation paradigm based definitions also. The property that we prove about the interrelations of different values generated in the execution of the multiparty protocol is enough for us to infer the privacy property of the protocol (which is along the same lines as several known definitions of security). More discussions relevant to the exact definition of security of the multiparty protocol for a.e.s.c. precede the definitions themselves.

In retrospective, what we have achieved here is a separation of what is real and concrete (the parties, protocol or some values etc.) from what is purely abstract (the simulator), in the process questioning and dissolving the belief about the existence of a "real" simulator and issues related to considering the feasibility of execution of the simulator in "reality" (for example by seeking "feasible" inputs to feed it with etc.). Stated simply, simulators do not "really" exist but relations like PPT computability or just computability between

different values generated in the execution of a multiparty protocol do and some properties of these relations are interesting from the perspective of cryptography. An alternate view of the definitions is presented in Subsection A.4 that the reader might find easier to digest but we emphasize the importance of the discussion in this section.

## D Realizing $\mathcal{T}$-Secure channels on incomplete networks

In this section we present multiparty protocols that utilize the infrastructure of incomplete networks to establish authentic or secure channels on the incomplete network. For the sake of completeness and readability we include again some of the properties that a network is required to possess so that these channels can be realized.

Let $N = (V, E, E_d)$ be an incomplete network. Let $\mathcal{A}$ be a $\mathcal{T}$-restricted adversary. A multiparty protocol $\gamma(p_u, p_v, N, s, c)$ realizes $\mathcal{T}$-secure channel between $p_u$ and $p_v$, if for all $\overrightarrow{C} \in \mathcal{T}$, $\gamma$ realizes an (almost) secure channel between $p_u$ and $p_v$ (Formal Definition 16). Dropping the privacy condition from $\mathcal{T}$-secure channel, we obtain the relaxed notion of $\mathcal{T}$-authentic channel (Corresponding formal Definition 17 appears in the Appendix).

### D.1 The message transmission protocol

In this subsection we present a multiparty protocol that establishes $\mathcal{T}$-secure channels between appropriate pair of nodes of an incomplete network $N$ when the adversary is $\mathcal{T}$-restricted. Let $N$ represent the communication infrastructure given to the set of parties $V$, with the Special Property $\mathcal{T}_{p_u,p_v,\beta,c}$.

**Definition 18.** *Network N possesses Property $\mathcal{T}_{p_u,p_v,\beta,c}$, if the following two conditions hold true:*

1. *For every $\overrightarrow{C} \in \mathcal{T}$, corrupted by $\mathcal{A}$, there exists a path of secure channels, connecting uncorrupted nodes, of length at most $c * \lg_2 n$ from $p_u$ to $p_v$.*
2. *There exists a multiparty protocol $\beta(p_u, p_v, N)$ that realizes $\mathcal{T}$-authentic channel between $p_u$ and $p_v$. (The channel established by $\beta$ is also referred to as the additional channel between $p_u$ and $p_v$.)*

*Remark 5.* The first condition of the special property can be relaxed so that it requires only a single path connecting honest nodes using secure channels - in which case the privacy condition would hold only for *completely* honest parties, whereas the above condition guarantees privacy against *semi-honest* parties, [CFGN96]. In [Vay06], it was shown how to construct a family of networks (with weakened parameters) for which two node-disjoint paths of faulty nodes are guaranteed, using another family of networks where only a single path of non-faulty nodes is guaranteed. However, as discussed in the last section its use was only for Upfal graphs, [Upf92], for which we have observed stronger guarantees, [BBC+06] (Please consult last section of Appendix).

We present a multiparty protocol $\gamma(p_u, p_v, N, s, c)$ to establish an (almost) secure channel between nodes $p_u$ and $p_v$ on network $N$, which possesses the above $\mathcal{T}_{p_u,p_v,\beta,c}$ Property: Several protocols can be employed for this purpose. The protocol for PSMT from [GO08] uses error correcting codes. The protocol presented here is based on elementary ideas for bit transmission, and is easily extended for message transmission. Roughly, the idea for the bit transmission protocol is the following: In Step 1 all paths of length at most $c * \lg_2 n$ between nodes $p_u$ and $p_v$ in network $N$ are used to send sequences of uniformly chosen random bits. In the rest of the protocol $\gamma(p_u, p_v, N, s, c)$, only $\mathcal{T}$-authentic channel(s) are used for exchanging messages between $p_u$, $p_v$. First, except for hiding a single bit from each rest of the block is revealed to the receiver via the public authentication channel. The receiver then intimates the sender of faulty channels which are excluded. The randomness contribution from each of the non-faulty channels is then XORRED with the message bit and sent to the received over the public authentication channel.

Now we present a multiparty protocol $\gamma(p_u, p_v, N, s, c)$ for node $p_u$ to send a single bit message to node $p_v$, securely using the infrastructure of network $N = (V, E)$. Let $l(n)$ be some polynomial in $n$, which will be defined according to the guarantee of security made for $\gamma$. We make a few highlighting remarks about the features of the multiparty protocol $\gamma(p_u, p_v, N, s, c)$:

1. Recall the definition of (physically realized) authentic and secure channels of network $N$, in Section 3. Each of these channels is parametrized by variable $r$, which refers to the number of rounds after which a message sent by the sender, along the channel, reaches the receiver.

2. Each *Step* of $\gamma(p_u, p_v, N, s, c)$ consists of multiple rounds. The first step consists of a total number of $(c * \lg_2 n) * r$ rounds, where the constant $c$ depends on network topology $N$. A message sent over the *additional channel* realized by $\beta(p_u, p_v, N)$ takes $r * r_{\beta,N}$ rounds.

3. The protocol $\gamma(p_u, p_v, N, s, c)$ runs for a total of $c * \lg_n * r + 5 * r_\beta * r + 2 * r$ rounds. No messages are exchanged in the last two rounds. The presence of constant $2 * r$ is due to technical reasons, to carry out higher level proofs of multiparty computation.

**MPC protocol $\gamma(p_u, p_v, N, s, c)$: Setup Phase.**

Enumerate all paths of length at most $c * \lg_2 n$ between $p_u$ and $p_v$ in network $N$. Let $PA = \{pa_1, pa_2, \ldots, pa_w\}$ be the set of paths, $w$ in number. The set $PA$ and the path-ids etc. are known to all the parties of the network as a result of the set up phase.

1. $p_u \rightarrow p_v$ : Along each path $pa_i \in PA$, $\forall i \in [n]$ between $p_u$ and $p_v$, send a block of $k(w,s)$ uniformly chosen random bits, for some polynomial $k(.,.)$ that will be specified later. (After forwarding the block to the next node on the path, the intermediate node deletes the block.)
2. $p_u \rightarrow p_v$ : For each of the $w$ paths, choose an integer $i_p \in \{1 \ldots k(w,s)\}$ uniformly at random and send this set of integers to the node $p_v$ over the *additional channel*, realized by multiparty protocol $\beta(p_u, p_v, N)$.
3. $p_v \rightarrow p_u$ : Node $p_v$ hides the $i_p^{th}$ bit from the respective block of random bits, and sends the rest of the block to $p_u$ over the *additional channel*.
4. $p_u \rightarrow p_v$ : Node $p_u$ identifies the *faulty* paths, as follows: $p_u$ checks for mismatch between the block created after removing the hidden bit (i.e., $i_p^{th}$ bit) from the respective block sent in Step 1, and the block sent in Step 3 (for the same path).
   If all the paths are identified as *faulty*, then $p_u$ sends a message **PROTOCOL-ABORT** to $p_v$ over the *additional channel*. Otherwise, let $b^* = \bigoplus_{j=1}^{g} b_{i_j}$ be the exclusive-or of the hidden bits from all the paths identified as non-faulty. The node $p_u$ sends the following to node $p_v$ over the *additional channel*:
   (a) The subset of the identities of all the paths identified as *non-faulty*.
   (b) If $m_o$ is the bit to be sent, send $b_o = m_o \oplus b^*$ over the *additional channel* to node $p_v$.
5. $p_v$: Let $b'_o$ be the bit received in the previous Step. $p_v$ receives the identities of all the "non-faulty" paths and computes the XOR of the "hidden" bits of the blocks sent over each of the paths in previous Step i.e., its own values of $b_{i_j}$'s and the bit $b'_o$ received in the previous Step.
   Node $v$ extracts the bit for each of the $l(s)$ parallel executions and accepts the majority as its output value.

We will conclude with the following theorem that characterizes $\gamma$. The proofs of correctness and privacy/simulatability are presented in the next two section D.2

**Theorem 6.** *Let $N = (V, E)$ be an incomplete network with Property 9 defined above. Then, protocol $\gamma(p_u, p_v, N, s, c)$ realizes a $\mathcal{T}$-secure channel between $p_u$ and $p_v$.*

### D.2 Simulation of $\gamma$

It is useful to be able to establish secure channels between distant nodes of an incomplete network so that these nodes can communicate with each other. And, this is interesting primitive to achieve by itself. But there is a more important and global view of the scheme of affairs i.e., namely the goal to realize secure multiparty computation on incomplete networks for which the simulation of $\gamma$ is found necessary. For this let us briefly revise that the way we achieve this is following: First we define a network corresponding to the incomplete network $N$ which has virtual channels connecting all ordered pairs of nodes that not already connected to each other via an edge in the original incomplete network. We then inductively replace such virtual channels channels by PSMT protocols, while proving security of intermediate protocols at every

intermediate step. Let's just work with the replacement of the first channel of the incomplete network (the rest will have similar reasonings/implications).

The simulator for the base case assigns random inputs to the honest unsacrificed parties and the relevant inputs (that it is provided with) for the sacrificed and corrupted parties. It is easy to see how the simulation is carried from here. Now when the first virtual channel is replaced by a PSMT, several cases arise: The directed channel simulated may be from a corrupted party to a corrupted party, or unsacrificed honest party or sacrificed party. Similar cases can exist when the sender is unsacrificed honest party or a sacrificed party. Thus total of 9 cases can exist. We will see that for some of the cases the simulator that we demonstrate for $\gamma$ may require inputs from outside and for other cases the simulator may not require any inputs from outside i.e., some higher level simulator that invokes this basic simulator for $\gamma$. Nevertheless, the simulator for $\gamma$ is given the input message to be transmitted whenever the sending party is an honest party, irrespective of whether it is sacrificed or unsacrificed. For the corrupted sender the message is generated by adversary so the simulation becomes trivial. For the honest parties, the message is always given to the simulator (it received from the simulator for higher level where the message is actually generated). Nevertheless, it is important to show that for channels which are secure the adversaries's view are indistinguishable for the two cases when a message $m$ or another message $m' \neq m$ is sent on the secure channel. This is important because even for the corresponding ideal functionality it is not possible to extract the message that was actually sent on the channel. It is important that the privacy of such a message be preserved from the adversary because should the adversary obtain enough number of such messages, then it will be able to extract the input of some honest party, say, which it should not have and which in turn would make the views of the adversary generated from the real execution of the protocol indistinguishable from the views of the adversary generated via simulation - leading the proof of privacy to fall apart.

Now, let's get back to where we were at. Namely, we are going to show simulation for all cases: When the sender is an honest sacrificed or unsacrificed party then the simulator will be given the message to be sent and when the sender is a corrupted party then the simulator is not given the message to be sent as this message is supposed to be generated for the adversary. These simulators will be invoked to complete simulation at the higher level. There is one more distinction we would need to make, namely that when the channel to be simulated is a secure chennal then the simulator will not be given any input but only the length of the message. Indistinguishability of the simulated view of the adversary from the one generated from real execution of the PSMT would then imply that the message was sent privately over the channel.

The proofs from here on are relatively straightforward except quite tedious technically. Now, let us prove the simulatability property of protocol $\gamma$ for all the different cases.

**Simulatability of adversaries view for multiparty $\gamma(p_u, p_v, N, s, c)$.** When the channel simulated by $\gamma(p_u, p_v, N, s, c)$ is to be proved secure, then the simulator is only given the length of the message and not the message. The simulator is given the message being transmitted on the channel for all other cases. We consider each case separately and qualify it accordingly.

*Claim.* Let multiparty protocol $\gamma(p_u, p_v, N, s, c)$ be executed on network $N$ with Property $\mathcal{T}_{p_u,p_v,\beta,c}$, 18.

There exists a simulator, that runs in time polynomial in running time of $\mathcal{A}$ and generates a distribution of views of $\mathcal{A}$ that is indistinguishable from distribution of views of $\mathcal{A}$ generated from executing $\gamma(p_u, p_v, N, s, c)$ on $N$, for all quadruplets $\overrightarrow{C} \in \mathcal{C}$ corrupted by $\mathcal{A}$.

*Proof.* Let $\mathcal{A}$ corrupt quadruplet $\overrightarrow{C} \in \mathcal{T}$. Depending on the subset of nodes corrupted by $\mathcal{A}$, we shall have several cases and the behavior of the simulator may be different for each case.

We shall divide the proof according to the choice of parties and channels corrupted by $\mathcal{A}$, and whether additional channel simulated by $\beta(p_u, p_v, N)$ is authentic or tamperable. In either case, there exists a simulator $Sim^{\beta}$ that generates a distribution of views of $\mathcal{A}$ that is indistinguishable from the views of $\mathcal{A}$ generated from real execution of multiparty protocol $\beta$. (In particular, as discussed above, when $p_u$ is honest then the simulator shall be given the input message to be sent by $p_u$ if the channel to be simulated is not secure and if $p_u$ is faulty then $\mathcal{A}$ itself is the generator of the message).

We consider all the cases:

1. Nodes $p_u, p_v$ are honest and there exists at least one path of non-faulty nodes of length at most $c * \lg_2 n$ between $p_u$ and $p_v$: In this case simulator $Sim^{\beta}$ is given as input the message to be sent.
   (a) $\overrightarrow{C} \in \mathcal{T}$ (Most difficult case: Secure channel) In this case we demonstrate a simulator that simulates the view of the adversary $\mathcal{A}$, but takes no input i.e. is a Zero-Knowledge simulator, thus proving perfect *Privacy* property for the established channel.
   **The simulator $\mathcal{S}$.** The simulator $\mathcal{S}$ is given as input $\overrightarrow{C}$ corrupted by the adversary $\mathcal{A}$ and whether the additional channel is authentic or tamperable. Note that for the MPC protocol $\gamma(p_u, p_v, N, s, c)$, only $p_u$ receives an input message and only $p_v$ outputs an output value, the rest of the parties just participate in the protocol $\gamma(p_u, p_v, N, s, c)$ without taking any inputs or producing any outputs. Next, also note that *the input message bit* is used only in the last Step of the protocol - the rest of the steps only involve exchanging uniformly random bits. The simulation of the protocol is quite straightforward from hereon.
   We are going to show the simulation of execution of one sequence of steps of protocol $\gamma(p_u, p_v, N, s, c)$. The extension to $l(s)$ independent parallel repetitions for each different message bit or repetition for amplification is a straightforward extension. Note, that in this case the simulator is given as input the length of the message.
   The simulator $\mathcal{S}$ simulates the role of all the honest parties, $p_u$ and $p_v$ as it is. In the first step the simulator chooses a sequence of uniformly chosen random bits for each path and simulates the sending of this sequence of uniformly chosen random bits along the path as follows: When an intermediate party sends a message to an adversary corrupted party, the simulator sends the message to $\mathcal{A}$ which may modify the message before passing it to the next honest party (which is simulated by the simulator). Then, $\mathcal{S}$ chooses the identity of one bit from each sequence of random bits and invokes the simulator $Sim^{\beta}$ for simulating sending this set of identities from $p_u$ to $p_v$ through an authentic channel. In the next step the simulator takes all the sequences, in their entire ties except for the hidden bits (in previous step) and invokes $Sim^{\beta}$ for simulating sending this set of identities from $p_u$ to $p_v$ through an authentic channel simulated by protocol $\beta$.
   In the next step the simulator identifies the faulty paths based on the corrupted blocks in first Step, and invokes the simulator $Sim^{\beta}$ for simulating sending the identities of these paths through an authentic channel simulated by protocol $\beta$.
   Now since we are given that there exists at least one non-faulty path in network $N$ between $p_u$ and $p_v$ we know that one at least random bit is completely hidden from the adversary - there may be more but at least one is completely hidden. The simulator computes the XOR of all of these hidden bits (chosen from all non-faulty paths) with $r'$ and invokes simulator $Sim^{\beta}$ on this input.
   [Simulation for *parallel* repetitions for the same bit or for different repetitions for different message bits is a straightforward extension - the proof follows by use of hybrid argument, la [GM82]]
   Observe that $Sim^{\beta}$ and hence simulator *Sim* run in time polynomial in the running time of the adversary $\mathcal{A}$.
   Proof of indistinguishability of $\mathcal{A}$'s view. Intuitively, the argument is as follows: During the simulation of the first few steps of the protocol only uniformly chosen random bits are used(exchanged between) the parties. Now, for the choice of $\overrightarrow{C} \in \mathcal{T}$ there is one path secure from the adversary. The uniformly chosen random bit from this path remains secure from $\mathcal{A}$ and is used to hide the message bit used in the last step of $\gamma$. But the XOR of a random bit with the message bit creates a ("random" looking) bit that is indistinguishable from a random bit. Thus, the view of $\mathcal{A}$ generated by the simulator or in the real execution of $\gamma$ is indistinguishable. Details follow.
   We show that the distribution of the views of $\mathcal{A}$ generated by the above simulator, which dozen't take any input, is indistinguishable from the distribution of the views of the adversary generated from execution of $\gamma(p_u, p_v, N, s, c)$ with some arbitrary message $m$.

For this we show that this is the case after simulation of every step of $\gamma$. First, notice that the adversary plays an active role in formulating its view only in the first step - after the first step of protocol $\gamma(p_u, p_v, N, s, c)$ the adversary is passive and just passively collects all messages (since the additional channel simulated by $\beta$ is authentic). We are going to show indistinguishability of the distribution of adversaries view after every step of $\gamma(p_u, p_v, N, s, c)$.

In the first step notice that the *probability* with which the adversary chooses to corrupt the blocks of uniformly chosen random bits sent on the faulty paths is the same whether its a real execution of $\gamma(p_u, p_v, N, s, c)$ or the simulation. In particular, the *probability* with which the adversary chooses to corrupt the blocks, sent on some particular paths, at some particular corrupted nodes and at some particular positions is exactly the same for the simulation of the ideal world as it is in the real execution of $\gamma(p_u, p_v, N, s, c)$ given that the blocks were chosen uniformly at random. Thus, the distribution of views of the adversary $\mathcal{A}$ generated after the real execution of first step of $\gamma(p_u, p_v, N, s, c)$ or after the simulation of the first step of $\gamma(p_u, p_v, N, s, c)$ by the simulator is exactly the same or a Distinguish er may be constructed using the adversaries program which can distinguish between two sources of uniformly random bits - contradiction.

Now notice that the adversary dozen't contribute actively in the constitution of its view, after the first step of the protocol, as it just receives all the messages exchanged between $p_u$ and $p_v$ on $\mathcal{T}$-authentic channel.

Fix a view $V_w$ of the adversary constituted up till Step 1. We know that the probability that this has been picked from such views generated from real execution of protocol $\gamma(p_u, p_v, N, s, c)$ or from ideal world simulation is equal i.e., $\frac{1}{2}$.

Observe also that for protocol $\gamma(p_u, p_v, N, s, c)$ every view of the honest parties generated till Step 1, that is compatible to the same view of the adversary generated till Step 1, is equally likely to be from real life execution of $\gamma(p_u, p_v, N, s, c)$ or from ideal world simulation. Thus we can fix a given view of the honest parties as well as the view of the adversary till Step 1 and this is equally likely to be from real world execution of protocol $\gamma(p_u, p_v, N, s, c)$ or from ideal world simulation.

Observe that once the view of the honest parties and the adversary is fixed for Step 1, if the same random choices are made for nodes $p_u$ and $p_v$ during Step 2 through Step 5 in real execution of $\beta$ as well as in simulation by the simulator $Sim^{\beta}$, the view of the adversary generated is identical.

So the distribution of the views of the adversary generated so far (i.e., up to Step 5 of $\gamma(p_u, p_v, N, s, c)$), constituted from real execution or ideal world simulation are identical.

The Step 6 needs a slightly different analysis because in this step messages are sent from node $p_u$ to $p_v$ which depend on the actual message $m$. This message $m$ is available to the node $p_u$ in the real world execution of $\gamma(p_u, p_v, N, s, c)$ but not to the simulator because the simulator is not given any input. Observe that for this case, that since there exists at least one path of non-faulty nodes between nodes $p_u$ and $p_v$ along which a block of uniformly random bits was sent in Step 1, therefore at least one random bit is secure from the adversary. This random is also used to hide the "message bit" in Step 6. Let this random bit be $r_0$. The adversaries view for this step would be $r_0 \oplus m_i$ for real world execution and a uniformly chosen random bit $r'$ for the simulation. However since $r_0$ is a uniformly chosen random bit completely hidden from the adversary (just like $r'$ is uniformly chosen random bit uncorrelated to the rest of the view of the adversary) and hence uncorrelated to the rest of the view of the adversary $r_0 \oplus m_i \approx r'$. This completes the argument for the indistinguishability of the distribution of the views of $\mathcal{A}$ generated from execution of one sequence of Steps 1 through Step 6, of protocol $\gamma(p_u, p_v, N, s, c)$.

To take into account the view of $\mathcal{A}$ constituted from $l(s)$ different parallel executions repeated, for each of the $r$ different message bits, the construction of a hybrid argument, a la [GM82], is straightforward.

Thus the distribution of the views of the adversary generated from the real world execution of $\gamma(p_u, p_v, N, s, c)$ is indistinguishable from the distribution of the views of the adversary generated from the ideal world simulation by the simulator.

(b) $\overrightarrow{C} \notin \mathcal{T}$ i.e., Channel simulated by $\beta$ between $p_u$ and $p_v$ is not secure.

In this case the channel established between parties $p_u$ and $p_v$ by protocol $\gamma(p_u, p_v, N, s, c)$ is not secure but may be arbitrarily *tamperable*. This could happen for several different reasons: (1) There is no path of perfectly secure channels between $p_u$ and $p_v$; (2) The public authentication channel is under the control of the adversary. All these cases fall under the case of a tamperable channel for which we need to prove the simulatability condition but not Zero-Knowledge simulatability.

This is seen as follows: In Step 1, $p_u$ may be able to send a block of uniformly chosen random bits to $p_v$, using the path of non-faulty nodes between $p_u$ and $p_v$, which may be perfectly hidden from the adversary. However, in the subsequent steps the adversary, which has full control over the additional channel, not only receives the messages sent by $p_u$ but has full control for the rest of the steps of the protocol. The adversary may be able to get $p_u$ and $p_v$ to agree on arbitrary message (or it may be possible that adversary does not get all non-faulty paths declared "faulty" by use of the public discussion channel and the random bits from at least one of the non-faulty paths is used for hiding the message being sent on the channel. For this case, although we prove no new properties about the privacy of the channel, the channel may behave as not eavesdroppable but corruptible. But except for showing the simulation (which gets the input from higher level simulator) we just do not care because we do not need to prove that the demonstrated simulator is not Zero-Knowledge because we do not need to claim this property for the proof of privacy).

Since we are not required to demonstrate a Zero-Knowledge simulator for this case we do not face any hardships. As before if $p_u$ is honest then the simulator gets the message from a higher level simulator which is simulating the party $p_u$ and the simulation becomes straightforward from after here on, else if $p_u$ is faulty then $p_u$ possess the message and carrying out the simulation poses no problems.

2. Node $p_u$ is corrupted. Recall that only $p_u$ is supposed to receive an input message for $\gamma(p_u, p_v, N, s, c)$. In this case $p_u$ itself is controlled by $\mathcal{A}$. In particular, the $\mathcal{A}$ itself possesses the message that $p_u$ intends to transmit to $p_v$, which can be honest or corrupted.

   First note that the simulator is only given the quadruplet $\overrightarrow{C}$ corrupted by $\mathcal{A}$ for this case. All we need to note is that the honest nodes only make uniformly random choices when/if they are required to make such choices during the execution of $\gamma(p_u, p_v, N, s, c)$. Otherwise, they are just limited to forwarding messages exchanged on the authentic channel. Thus the simulator does not need any information or message for simulating the role of the honest nodes in this case - it is the adversary who needs the message and it also possesses the message. The adversary controlled $p_u$ may not execute the protocol as required and abort or send unexpected/arbitrary messages but all this is perfectly simulated.
3. Otherwise. This means that (a) $p_u$ is honest and, (b) That either, (1) $p_v$ is corrupted or, (2) There exists no path of non-faulty nodes of length at most $c * \lg_2 n$ between $p_u$ and $p_v$ in network $N$ or, (3) The channel realized by $\beta$ is not authentic. In either of the three cases *we will not be required to prove that the protocol* $\gamma(p_u, p_v, N, s, c)$ *realizes a secure/private channel between nodes* $p_u$ *and* $p_v$ *for this case*. Since, we are not required to prove that the channel realized by $\gamma$ is not private it is enough for us to demonstrate a simulator that takes as input the message $m$ that $p_u$ intends to send to $p_v$. Depending on the other conditions full or no information about the message $m$ may get leaked to the adversary just as it would in the execution of protocol $\gamma(p_u, p_v, N, s, c)$. For this case, we demonstrate a simulator that is given as input the duple $\{m, \overrightarrow{C}\}$, which will be sufficient for the the higher level simulator. But construction of such a simulator is quite straightforward, as noted above, because once the simulator possesses the only input any party is given in the execution of $\gamma$, *the simulator just simulates the role of all honest parties "exactly" as in the real execution of protocol* $\gamma$, while sending the messages to be sent to the corrupt parties to the adversary and vice-versa receiving too. If the simulator makes the same random choices for the honest parties and the adversary makes the same random choices for the corrupted parties, for the ideal world simulation as well as real world execution of protocol $\gamma(p_u, p_v, N, s, c)$, then the adversary is going to end up with exactly the same view. Hence the distributions of the views of the adversary for the

two cases would also be indistinguishable. Otherwise, a contradiction would result due to the existence of a distinguish er that distinguishes between two sources of uniformly chosen random bits.

### D.3 Correctness of protocol $\gamma$

We are going to show that protocol $\gamma(p_u, p_v, N, s, c)$ described above realizes a $\mathcal{T}$-secure channel between $p_u$ and $p_v$.

**Theorem 7.** *Let $N = (V, E)$ be an incomplete network with Property 18 defined above. Then, protocol $\gamma(p_u, p_v, N, s, c)$, executed on $N$, realizes a $\mathcal{T}$-secure channel between $p_u$ and $p_v$.*

*Proof.* Recall the definition of $\mathcal{T}$-secure and $\mathcal{T}$-authentic channels from Section 3. We shall show that channel realized by $\gamma(p_u, p_v, N, s, c)$ is secure:

1. Privacy: We demonstrate a simulator that runs in time polynomial in the running time of adversary $\mathcal{A}$, such that for every input vector $\overrightarrow{I}$, such that $\forall j \in [n], j \neq u : I[j] = \phi$ and $I[u] = m \in \{0,1\}^r$ (here $m$ is the message to be sent from $p_u$ to $p_v$), subset $C$ of nodes corrupted by the adversary $\mathcal{A}$ such that $C \in T$, the simulator *Sim* takes as input the subset $C$, network topology $N$ and generates a distribution of views of $\mathcal{A}$ that is indistinguishable from the distribution of views of $\mathcal{A}$ generated from real execution of $\gamma(p_u, p_v, N, s, c)$.
   The existence of such a simulator follows as a corollary to Theorem D.2. In particular we are considering case 1a, for which the simulator does not take $\overrightarrow{I}$ as the input. This completes the proof of privacy.
2. (almost) Correctness: It is given to us that Protocol $\beta(p_u, p_v, N)$ realizes a $\mathcal{T}$-authentic channel from $p_u$ to $p_v$.
   We have to prove that the channel realized by $\gamma(p_u, p_v, N, s, c)$ is (almost) correct when $\mathcal{A}$ chooses to corrupt quadruplet $\overrightarrow{C} \in \mathcal{T}$. For this just observe that $N$ possesses the property, 18: Thus for each $\overrightarrow{C} \in \mathcal{T}$, the protocol $\gamma(p_u, p_v, N, s, c)$ is equivalent to the bit transmission protocol, D.1.
   We shall show a simple lower bound on the probability of $p_u$ and $p_v$ agreeing on the same bit. We shall do it in two steps. First, we shall prove a lower bound on the probability of the event that all the channels chosen to hide bit 'b' in Step 4 are good i.e., were not manipulated by the adversary. Then, we shall do a standard probability amplification using Checkoff's bound.
   Define $X_i$ to be the event that Channel $id_i$ was faulty and chosen. Correspondingly, define $X_i$ for all $i \in [N]$. Then, the probability of choosing at least one faulty channel is $Pr[X = \bigcup_{i \in N} X_i]$. This is upper bounded by the Union bound, $Pr[X = \cup_{i \in N} X_i] \leq \sum_{i=1}^{i=N} Pr[X_i]$.
   The probability that a faulty channel $id_i$ was chosen for transmission is $Pr[X_i] \leq \frac{1}{k(N,s)}$. Thus, in the worst case we have $Pr[X = \bigcup_{i \in N} X_i] \leq \frac{N}{k(N,s)}$ as the upper bound of choosing at least one faulty channel. The probability that all the channels chosen are good is $\frac{Pr[}{lineX]}$, $\frac{Pr[}{lineX] \geq 1 - \frac{N}{k(N,s)}}$
   In this case we are assured of a good transmission i.e., the minimum probability of a good transmission is $1 - \frac{N}{k(N,s)}$. If $k(N,s)$ is chosen to be $N * f(s)$ for some function $f(.)$, then this expression evaluates to $\frac{Pr[}{lineX] \geq 1 - \frac{N}{k(s)} \geq 1 - \frac{1}{f(s)}}$
   The probability that $p_v$ receives incorrect bit in Step 5, $Pr[X]$, is at most $\frac{1}{f(s)}$. $p_u$ and $p_v$ repeat Step 1 through Step 5, $l(s)$ times and take the majority value as the answer. By Checkoff's bound, the probability that $p_u$ and $p_v$ agree on dissimilar bits is: $Pr(b' \neq b) \leq exp(-\mu * \frac{\delta^2}{2})$, where $\delta = \frac{1}{2} - \frac{1}{2*(f(s)-1)}$ and $\mu = l(s) * (1 - \frac{1}{f(s)})$.
   The transmitted bit is correct with probability at least $1 - \exp^{\frac{-(f(s)-2)^2 * l(s)}{8 * f(s) * (f(s)-1)}}$. For $f(s) = 3$ and $l(s) = s$, the probability of transmitting a bit correctly is at least $1 - exp^{O(s)}$.

The analysis of Correctness condition is the same as that of the bit transmission protocol in, with $N = w, s = l$ channels and other suitable parameters for security.
The above analysis is to send a single bit message over the channel. To send message $m = (m_0, m_1, \ldots, m_o, \ldots, m_{|m|})$ (Where $m_i$ represents the $i^{th}$ bit of $m$) from node $p_u$ to node $p_v$, execute Step 1 through Step 5, *parallel* for each bit, $l(s)$ times concurrently. The $o^{th}$ bit computed in this manner is accepted as the $m_o^\gamma$ bit of message $m$.
If the message $m^u$ sent by $p_u$ and the message $m^v$ received by $p_v$ are different with a non-negligible probability, then by a straightforward hybrid argument at least one bit of the message must differ with a non-negligible probability, contradicting the above analysis.

From the above we have that multiparty protocol $\gamma(p_u, p_v, N, s, c)$ realizes a $\mathcal{T}$-secure channel between $p_u$ and $p_v$.

## E $\mathcal{T}$-secure computation on complete networks

In this section we shall prove that if there exists an unconditionally secure multiparty protocol in the vanilla model, which allows only for corruption of parties, according to Definition 12, then there exists an unconditionally secure multiparty computation protocol executed on a complete network in the model that allows for corruption of parties as well as channels, actively as well as passively, according to Definition 14.

Let $f$ be an $n$-variate function and $\Pi$ be an unconditionally secure multiparty protocol for evaluating function $f$ as long as at least $\lfloor \frac{2*n}{3} \rfloor + 1$ parties are honest. The following claim is well known:

*Claim.* There exist multiparty computation protocol that securely evaluates $n$-variate function $f$ according to Definition 12.

The three distinguishing features of Definition 14 with respect to Definition 12 are that in Definition 14: (1) Adversary $\mathcal{A}$ is allowed to corrupt parties as well as channels, passively as actively, so an adversary structure $\mathcal{T}$ is introduced, Definition 4 (2) The multiparty computation protocol assumes only an incomplete communication network that connects the parties. (3) Guarantees only statistical correctness and not perfect correctness.

Let $N_C$ be a complete undirected network.

**Theorem 8.** *There exists a multiparty computation protocol $\Pi'$ that $\mathcal{T}$-securely evaluates function $f$ according to Definition 14, on a complete network $N_C$ for all adversaries $\mathcal{A}$ restricted to a feasible adversary structures $\mathcal{T}$.*

*Proof.* Let $\Pi$ be a multiparty protocol as per Claim E. We show that the protocol $\Pi' = \Pi$, $\mathcal{T}$-securely evaluates function $f$ on complete network $N_C$, as according to Definition 14, for all feasible adversary structures $\mathcal{T}$.

Let $\overrightarrow{C} \in \mathcal{T}$. Since $\overrightarrow{C}$ is a feasible quadruplet of corruptions, then there exists a subset of honest parties $H \subset P$, $|H| \geq \lfloor \frac{2*n}{3} \rfloor + 1$, such that $\forall p_u, p_v \in H : (p_u, p_v) \notin \overrightarrow{C}[2] \bigcup \overrightarrow{C}[3]$.

We need to make sure that executing protocol $\Pi'$ on $N_C$, while the adversary $\mathcal{A}'$ corrupts $\overrightarrow{C}$, achieves the correctness and privacy condition of Definition 14.
For Correctness: It is enough to show that for every strategy of $\mathcal{A}'$ that corrupts quadruplet $\overrightarrow{C}$ as above, on execution of protocol $\Pi'$ on $N_C$, there exists a corresponding $\mathcal{A}$ that corrupts parties in subset $P - H$ while parties execute $\Pi$, such that the (distribution of the) view of the honest parties is indistinguishable for the two cases. This strategy of adversary $\mathcal{A}$ is as follows: The adversary $\mathcal{A}$ internally simulates $\mathcal{A}'$ for all the maliciously corrupted parties (i.e., those belonging to the subset $\overrightarrow{C}[0]$), simulates $\mathcal{A}'$ for all the passively corrupted parties (i.e., those belonging to the subset $\overrightarrow{C}[1]$) and internally simulates the protocol $\Pi'$ for all the remaining sacrificed honest parties (i.e., parties belonging to the subset $P - H - \overrightarrow{C}[0] - \overrightarrow{C}[1]$) and

appropriately simulates $\mathcal{A}'$ for passively and actively corrupted channels interconnecting the parties from subset $H$ to the parties in subset $P-H$ and transmits messages on other (uncorrupted) channels connecting to $H$ from $P-H$.

It is easy to verify that the distribution of the views of the (unsacrificed) honest parties $H$ are indistinguishable for the two cases and hence the distribution of transcripts of these parties at the end of the input commitment phase are indistinguishable. Therefore, it may be easily verified that $reveal_{\Pi'} = reveal_{\Pi}$, will achieve the characterization of the input commitment phase of protocol $\Pi'$. Now conditioned to the fact that the input values actually committed to by the parties are the same for the two cases, the output value is same and the (unsacrificed) honest parties receive identical output values for the two cases. The correctness of protocol $\Pi'$ then follows from the correctness of protocol $\Pi$, which satisfies the Definition 12 as according to the Claim E.

For Privacy: For the privacy condition it is enough to demonstrate an appropriate simulator $S'$. The simulator is given the starting input values and the input values committed to by the parties in subset $P-H$. By Definition 12, there exists a simulator $S$ that generates the distribution of views of the parties in subset $P-H$ (which is of size at most $\lfloor \frac{n}{3} \rfloor$) indistinguishably. The same simulator, slightly modified as described next, is invoked here as well. The view of $\mathcal{A}'$ generated by the corrupted channels in the new case, "looks" enhanced but is just a trivial extension because the simulator $S'$ is given the input and output values of all the parties in subset $P-H$ and simulates the sacrificed honest parties by itself.

If the distributions of views of $\mathcal{A}'$ generated by the above simulator, and the distribution of views of $\mathcal{A}'$ generated during the real execution of protocol $\Pi'$ have a non-negligible difference, then it translates to a non-negligible difference in the corresponding distributions generated, for $\mathcal{A}$, by simulator $S$ for proving the privacy property as according to Definition 12 - contradiction.

Finally, it is easy to verify the correctness of commitment phase for protocol $\Pi'$ for the new set up, as according to Definition 14, if original protocol $\Pi$ satisfies the correctness of commitment phase as according to Definition 12 (Basically, the function $reveal()$ translates is same and the difference is only in weakening the requirement from all honest parties being able to successfully commit to their initial input values, to at least $\lfloor \frac{2*n}{3} \rfloor + 1$ of the honest parties being able to commit to their initial input values, except for negligible probability).

*Remark 6.* We note again that the simulator $S$ and simulator $S'$ is required to produce appropriate distribution of the views only for the case when the input values, committed input values and the output values are the same as for the real execution of the protocol. In particular, the (unsacrificed) honest parties are always able to commit to their given input values, the sacrificed honest parties and the corrupted parties may not necessarily commit to their original input values which they start with. However, we discard simulations when the committed input values for corrupted and sacrificed parties are different then given to the simulator (What values have been committed to by the corrupted and sacrificed parties in the commit phase is verified at the end of the first phase of the protocol as the simulator can compute the committed input values for these parties from its own view) and are only required to produce the simulation for the case when the input values committed to by these (corrupted and sacrificed) parties is the one given as only that case is required to correspond to the output values - given to the simulator. Therefore, we only consider the simulated output distribution of the views of the adversary $\mathcal{A}'$ conditioned to the above fact.

## F Almost everywhere secure multiparty computation

In this section we present the full proof of the main Theorem 1. In general, a formal proof of a claim or a lemma has been preceded by intuitive discussion. For the sake of completeness we shall state the main theorem again. The theorem says that if network $N$ possesses, $\mathcal{T}_{\beta_{N},c}$-communicability property, then there exists a secure multiparty computation protocol $\Pi_N$ that $\mathcal{T}$-securely evaluates function $f$ on network $N$:

**Theorem 9.** *If network $N$ possesses $\mathcal{T}_{\beta_N,c}$-Communicability Property, 10, then there exists a multiparty computation protocol $\Pi_N$, that $\mathcal{T}$-securely evaluates function $f$, on network $N$, as according to Definition 14.*

*Proof.* The proof of this theorem is delicate for the primary reason that we need a protocol to realize secure channels between distant nodes of the network $N$ using the infrastructure of the network, to simulate any traditional information theoretic multiparty protocol, e.g. BGW, but are not allowed to use any protocol composition theorems.

At a very high level construction of the multiparty protocol and the proof of its security goes as follows:

1. Let $C_N = (V, E, E_{C_N})$ be the network on set of vertices $V$ constructed by connecting all the ordered pairs of nodes not already connected in network $N$, by *virtual* edges (Note, that since the virtual edges are directed, so for each $(u,v) \in V * V - E$ two directed edges with opposite directions are added to $E_{C_N}$). Now fix an arbitrary order on all the (directed) virtual edges, in subset $E_{C_N}$, and proceed by induction as follows:

   (a) Base case: We start by describing an adversary structure $\mathcal{T}_{C_N}$ for a complete network $C_N$ as a function of adversary structure $\mathcal{T}$, network $N$, and $\mathcal{T}_{\beta_N,c}$-communicability property, 10.
   Then, we show that there exists a multiparty protocol $\Pi_{C_N}$, 13, to be executed on the complete network $C_N$, that $\mathcal{T}_{C_N}$-securely evaluates $f$, as according to Definition 14.

   (b) By induction hypothesis, we are given an intermediate network $N_i$ for which we are given a multiparty protocol $\Pi_{N_i}$, such that protocol $\Pi_{N_i}$, $\mathcal{T}_{N_i}$-securely evaluates $f$, on network $N_i$, as according to Definition 14.
   We show construction of multiparty protocol $\Pi_{N_{i+1}}$, that $\mathcal{T}_{N_{i+1}}$ securely evaluates function $f$ on network $N_{i+1}$ (where network $N_{i+1}$ is network $N_i$ with the $i^{th}$ virtual edge $e_i$ removed from the network $N_i$ and adversary structure $\mathcal{T}_{N_{i+1}}$ is the same as adversary structure $\mathcal{T}_{N_i}$ except that edge $e_i$ is not present in adversary structure $\mathcal{T}_{N_{i+1}}$. This is because there is no edge $e_i$ in network $N_{i+1}$), as according to the Definition 14.

   (c) Lastly, note that the inductive hypothesis is applied till there are no virtual edges in the network that can be replaced by simulated channels, i.e., till $N_j = N$ and $\mathcal{T}_{N_j} = \mathcal{T}$, where $\mathcal{T}$ is the original adversary structure which allows network $N$ to possess certain communicability property. The multiparty protocol $\Pi_N = \Pi_{N_j}$ will $\mathcal{T}$-securely evaluate function $f$ on network $N$, as according to Definition 14.

Basically, at each step we shall replace a virtual channel, by a simulated channel that does precisely the same function[7] using the infrastructure of network $N$ i.e., is *authentic* if the original corruption makes it *authentic*, is *secure* if the original corruption is *secure* and is *tamperable* if the original edge is *tamperable*. Loosely speaking, the I. H. says that if we can realize $\mathcal{T}_{N_i}$-secure computation on incomplete network $N_i$ with $r$ real, $v$ virtual and $s$ simulated channel, then we can realize $\mathcal{T}_{N_{i+1}}$-secure computation on network $N_{i+1}$ with $r$ real, $v-1$ virtual and $s+1$ simulated edges.

Following will be the **outline of the rest of the proof**:

1. Ordering of the edges: We shall define an arbitrary ordering of the edges not present in the network $N$.
2. Round structuring of the protocols
3. Construction of the sequence of networks $N_i$
4. Construction of Adversary structures $\mathcal{T}_{N_i}$

[7] Actually, this is *almost precisely*, because with at most negligible probability the $\mathcal{T}'$-secure channel may fail to deliver the message and the originally perfect edge is replaced by a statistically perfect edge.

5. Statement and Proof of the Base Case
6. Inductive hypothesis discussion and formalization
7. Proof of Inductive hypothesis

**An ordering of** *virtual channels***:** Consider the subset of edges that are not present in the network $N$. These edges will be referred to as virtual edges and this subset of edges is referred to as $E_v = \{(p_u, p_v) | (p_u, p_v), (p_v, p_u) \notin E and (p_u, p_v) \notin E_d\}$. Now, consider an arbitrary ordering of these virtual edges belonging to subset $E_v$, $e_0^v = \phi, e_1^v = (p_{u,1}, p_{v,1}), e_2^v = (p_{u_2}, p_{v_2}), \ldots, e_l^v = (p_{u_l}, p_{v_l})$, such that $l = |E_v|$.
**Construction of network** $N_i, \forall i \in \{0, \ldots, l-1\}$**:** Network $N_0 = C_N = (V, E, E_d \bigcup E_v)$. Network $N_i$ is defined recursively in terms of network $N_{i-1}$ as follows: $N_i = (V, E, E_i = E_{i-1} - e_i^v)$.
**Adversary structure** $\mathcal{T}_{N_i}$**:** The intuition behind the construction of adversary structures for each $i$ is as follows:

Let $\gamma(p_u, p_v, N, s, c)$ be a multiparty protocol used to establish a channel from $p_u$ to $p_v$ for $(p_u, p_v) \notin E_i$ using the infrastructure of network $N$. Depending on the choice of $\overrightarrow{C}$ corrupted by the adversary, the channel established by $\gamma(p_u, p_v, N, s, c)$ may be (almost) secure, (almost) authentic or tamperable.

Thus, the tuplet $(\gamma(p_u, p_v, N, s, c), \overrightarrow{C})$ essentially determines the type of channel established between $p_u$ and $p_v$ on network $N$. An alternative way to look at this is to let edge $(p_u, p_v)$ always exist in the network, but let the adversary structure be enhanced depending on the type of channel realized by $\gamma(p_u, p_v, N, s, c)$ from $p_u$ to $p_v$. So, if on corruption of quadruplet $\overrightarrow{C}$, $\gamma(p_u, p_v, N, s, c)$ realizes an (almost) secure channel, then the new adversary structure contains the same $\overrightarrow{C}$. However, if on corruption $\overrightarrow{C}$, $\gamma(p_u, p_v, N, s, c)$ realizes an (almost) authentic channel from $p_u$ to $p_v$, then this is equivalent to assuming a secure channel between $p_u$ and $p_v$, passively corrupted by $\mathcal{A}$ i.e., in the new adversary structure $\overrightarrow{C}$ is enhanced to include $(p_u, p_v)$ i.e., $(p_u, p_v) \in \overrightarrow{C}[2]$ in the new adversary structure $(p_u, p_v)$. If the channel realized by $\gamma(p_u, p_v, N, s, c)$ is tamperable, then $(p_u, p_v)$ is added to $\overrightarrow{C}[3]$, for each $\overrightarrow{C}$. This is how the adversary structure is constructed for the base case network $N_0 = C_N$, which has all the virtual edges of subset $E_v$.

To define the adversary structure for the rest of the networks we work backwards from $\mathcal{T}_{C_N}$. But this is straightforward as the network $N_i$ does not possess the edge $e_i^v$ which is present in the network $N_{i-1}$ and is identical otherwise. So the virtual channel $e_i^v$ is not present in the corresponding adversary structure for network $N_i$ i.e., $\mathcal{T}_{N_i} = \{\overrightarrow{C} | \exists \overrightarrow{C}' \in \mathcal{T}_{N_{i-1}} s.t. \overrightarrow{C}[0] = \overrightarrow{C}'[0], \overrightarrow{C}[1] = \overrightarrow{C}'[1], \overrightarrow{C}[2] = \overrightarrow{C}'[2] - e_i^v, \overrightarrow{C}[3] = \overrightarrow{C}'[3] - e_i^v\}$
**Round structuring of multiparty protocols:** Our goal is to adapt a multiparty computation protocol that satisfies Definition 14, for a complete network to a multiparty computation protocol that is to be executed on an incomplete network, that possess some special communicability properties, and satisfies definition 14.

Even if we assume that there exists a multiparty protocol that establishes (almost) secure channel between some ordered pair of nodes of the incomplete network there arises the following *synchronization* issue with respect to using it for realizing secure function evaluation: Suppose it takes $\alpha$ rounds to send a message from party $p_u$ to party $p_v$ along a channel (realized via physical infrastructure) of the incomplete network. To send messages between distant nodes of the network, multiparty protocol $\gamma(p_u, p_v, N, s, c)$ is executed for this purpose and may run in $f_\gamma(\alpha, N)$ rounds, for some function $f_\gamma$, to deliver a message from party $p_u$ to party $p_v$. So we need to synchronize message transmissions to execute the original protocol for this we need to stretch an original round to several rounds.
**Initial Tentative Blow up:**
Consider the multiparty protocol $\Pi_{ec}$ from Theorem 8, which is restructured to run on the complete network, $C_N$ described above, as follows: Each Step of protocol $\Pi_{ec}$ is now expanded to $f_\gamma(\alpha, N)$ rounds. If in protocol $\Pi_{ec}$ the message was delivered between two parties in one round, now it is delivered in $\alpha$ rounds or $f_\gamma(\alpha, N)$ rounds depending on whether the message is sent along a *real channel* or a *virtual channel*. This is explained below:

**Definition 19.** *For a real channel, $F^r(S,R,edge-id)$, $r=\alpha$.*

*For a virtual channel, $F^r(S,R,edge-id)$, $r=f_\gamma(\alpha,N)$, for some function $f_\gamma$ (which is defined as the maximum number of rounds taken by the multiparty protocol $\gamma$ over all pairs of $p_u$,$p_v$ for network $N$).*

**Double blow up:**
We need another level of blow up in the total number of rounds. This blow up comes from needing to allocate a separate "slot" of "rounds for transmission" to each ordered pair of nodes of the network. This slotting is necessary as the view of the adversary generated from sending messages on one virtual channel influences the view of the adversary generated from sending messages on another virtual channel and these views can affect each other in the way the adversary chooses to corrupt messages on the other channel. Thus, slotting of a super-round for message transmission has been done to keep a cleaner account of the view of the adversary and for general tidiness of the proof.

More specifically, each round of the original multiparty protocol is now referred to as a *step* or a *super-round* and shall consists of two *phases*. Intuitively, the first phase of the super-round shall correspond to transmissions by honest (or passively corrupted) parties and the second phase shall correspond to transmissions by maliciously corrupted parties. This technicality is required to enable "rushing" adversary which is given the flexibility to first see the message on other channels, before deciding what messages to send on other channels.

Each phase will consist of $(\alpha+1)+n^2*f_\gamma(\alpha,N)$ rounds. Now, each phase of the multiparty protocol will be broken down into several slots ordered as follows: one slot of $(\alpha+1)$ rounds, followed by $n^2$ slots of $f_\gamma(\alpha,N)$ rounds each. The idea being that each real or virtual channel between any pair of nodes will have two unique slots to which it will correspond to, on which a message can be transmitted. Furthermore, no "real" activity as far as transmission of the original (like BGW, CCD) multiparty protocol messages is concerned will correspond to other rounds/slots which are not allotted to a given edge/channel. The real edge $(i,j)$ will correspond to the first slot (of $(\alpha+1)$ rounds) and the $n^2+2^{th}$ slot (of $(\alpha+1)$ rounds). The virtual edge $(i,j)$ will correspond to the $1+n*i+j^{th}$ slot (of $(f_\gamma(\alpha,N)$ rounds) and $2+n^2+n*i+j^{th}$ slot (of $f_\gamma(\alpha,N)$ rounds).

Traditionally, the advantage provided to the adversary by letting it see the messages from other parties before letting it decide what message to send is called "rushing". And, the need for division of each super-round into two phases and allocation of two slots for each directed channel is to capture this advantage given to the adversary in a clean and clear cut manner and is easily understood by the following easy to see claim:

*Claim.* Let $\mathcal{T}$ be a feasible adversary structure. Then, there exists a multiparty protocol $\Pi'$ that $\mathcal{T}$-securely evaluates $n$-variate function $f$, on network $N$, with $r_e$ real and $n^2-1-r_e$ virtual edges.

Furthermore, the message transmission on all the real and virtual edges is scheduled for $(\alpha+1)+n^2*f_\gamma(\alpha,N)+(\alpha+1)+n^2*f_\gamma(\alpha,N)$ slots, as discussed above. And, if $p_u$ is an honest node then it transmits a message during its allotted slot in the first phase of the super-round and if $p_u$ is a corrupted node then it may transmit its message during its slot in the second phase: of the super-round (The nodes do not distinguish between honest and corrupted nodes based on which slots other nodes transmit).

It is easy to see that if the protocol is secure against an adversary whose corrupted parties transmit only during their slots corresponding to the second phase of the super-round, then the protocol is secure against an adversary whose corrupted nodes transmit during their slots corresponding to the first phase or the second phase of the super-round. Thus it suffices to consider security against only the former types of adversaries (As the other simpler to emulate cases follow from this case).

We begin with the main proof, considering the Base case first.

**Base case for network $N_0=C_N$:** For the base case we have the complete network $N_0$. As discussed above, all the edges in $N_0$ which are also present in the incomplete network $N$ are referred to as *real* channels, while the rest of the edges are referred to as *virtual* channels.

The adversary $\mathcal{A}$ is restricted to adversary structure $\mathcal{T}_{C_N}$, as described above.

We are given that network $N$ possesses the $\mathcal{T}_{\beta_N,c}$-Communicability Property, 10. Because of the Communicability Property 10 of network $N$, the adversary structure $\mathcal{T}_{C_N}$ is feasible. By Theorem 8, there exists a multiparty protocol $\Pi_{ec}$, that $\mathcal{T}_{C_N}$-securely evaluates function $f$, on network $C_N$, as according to Definition 14, for every feasible adversary structure $\mathcal{T}_{C_N}$.

We adapt the multiparty protocol $\Pi_{ec}$ to the round structure F, as described above It is easy to see that the new multiparty protocol $\Pi_{N_0}$, executed on network $N_0$ (where corrupted and honest parties send messages as according to the round structure as described above 19), $\mathcal{T}_{N_0}$-securely evaluates function $f$, as according to Definition 14 (where $\mathcal{T}_{N_0} = \mathcal{T}_{C_N}$).

**Statement of Induction Hypothesis.** Let $\Pi^i$ be a multiparty protocol, to be executed on network $N_i$, that $\mathcal{T}_{N_i}$-securely evaluates function $f$, as according to Definition 14. Then, there exists a multiparty protocol $\Pi^{i+1}$, to be executed on network $N_{i+1}$, that $\mathcal{T}_{N_{i+1}}$-securely evaluates function $f$, as according to Definition 14 (where the construction of network $N_{i+1}$ and the adversary structure $\mathcal{T}_{N_{i+1}}$ has been discussed above).

**Proof of the Inductive Hypothesis:** First note that the only difference between network $N_i$ and $N_{i+1}$ is that in network $N_i$ there is a virtual channel $e_i^{u,v} = (p_u, p_v)$ between $p_u$ and $p_v$, which is not present in network $N_{i+1}$. Similarly, the only difference in adversary structure $\mathcal{T}_{N_i}$ and $\mathcal{T}_{N_{i+1}}$ is that edge $e_i^v$ is not *passively* or *actively* corrupted in adversary structure $\mathcal{T}_{N_{i+1}}$, as in fact it does not even exist in network $N_{i+1}$.

We shall show how to adapt protocol $\Pi^i$ to be executed on network $N_{i+1}$ to protocol $\Pi^{i+1}$ to be executed on $N_{i+1}$.

*Description of* $\Pi^{i+1}$*:* The multiparty protocol $\Pi^{i+1}$ is exactly the same as protocol $\Pi^i$ except for the following difference which corresponds to the difference in the underlying topology of the network $N_{i+1}$ on which it is to be executed, with respect to network $N_i$: The virtual edge $e_{i+1}^v$ in the network $N_i$ does not belong to network $N_{i+1}$. This translates to the following difference in protocol $\Pi^{i+1}$ with respect to $\Pi^i$: When message $m$ is to be sent on the virtual channel $e_{i+1}^{u,v} = (p_u, p_v)$, the parties execute protocol $\gamma(p_u, p_v, N, s, c)$ instead of sending it on the virtual channel $e_{i+1}^{u,v}$, which does not exist in the network $N_{i+1}$, in their appropriate slot of the appropriate phase, as according to the round structuring described in F. Now, we prove the following claim about multiparty protocol $\Pi^{i+1}$:

*Claim.* Multiparty protocol $\Pi^{i+1}$, $\mathcal{T}_{N_{i+1}}$-securely evaluates function $f$ on network $N_{i+1}$, as according to Definition 14.

*Proof.* We shall prove that the multiparty protocol $\Pi^{i+1}$, $\mathcal{T}_{N_{i+1}}$-securely evaluates the function $f$, as according to Definition 14.

First, review the the difference in adversary structure $\mathcal{T}_{N_i}$ with respect to the adversary structure $\mathcal{T}_{N_{i+1}}$, as discussed above.

We shall demonstrate an adversary $\mathcal{A}_i$ that attacks multiparty protocol $\Pi^i$'s execution on network $N_i$, such that protocol $\Pi^{i+1}$ $\mathcal{T}_{N_{i+1}}$-secure evaluates function $f$ on network $N_{i+1}$, as according to Definition 14, if protocol $\Pi^i$ $\mathcal{T}_{N_i}$-securely evaluates function $f$ on network $N_{i+1}$. For this we have to prove the following two properties of protocol $\Pi^{i+1}$:

1. Correctness: To prove the correctness condition of protocol $\Pi^{i+1} = (\Pi^{i+1}(1), \Pi^{i+1}(2))$ we need to prove the correctness of the commitment phase $\Pi^{i+1}(1)$ and the correctness of the computation phase $\Pi^{i+1}(2)$ of the protocol.
   The correctness of the commitment phase $\Pi^{i+1}(1)$ of protocol $\Pi^{i+1}$ is straightforward, and follows from observing that same function reveal() associated with the commitment phase of protocol $\Pi^i$ has the requisite properties for protocol $\Pi^{i+1}(1)$ also.
   For the correctness for the computation phase, we need to prove that for the vector of input values $\overrightarrow{x}$ committed to by the parties, the output of the honest parties is $f(\overrightarrow{x})$, except for negligible probability.[8]

[8] Note, that only statistical correctness is required for this phase because the channel simulated by protocol $\gamma$ are (almost) secure and not perfectly secure. Hence, replacing one perfectly secure channel by an (almost) secure channel would allow negligible

We prove the correctness condition by a reduction argument. In particular, we prove that protocol $\Pi^{i+1}$ is correct if protocol $\Pi^i$ is correct. For this we describe an adversary $\mathcal{A}_i$ that attacks (execution of) protocol $\Pi^i$ as follows:

Adversary $\mathcal{A}_i$ internally simulates $\mathcal{A}_{i+1}$ by providing it with all the messages that corrupted parties receive from the honest parties during the execution of protocol $\Pi^i$. Furthermore, $\mathcal{A}_i$ sends the messages generated by these corrupted parties to be sent to the respective honest parties. Corruption of channels is handled in a similar manner, where $\mathcal{A}_i$ just hands the respective messages to $\mathcal{A}_i$ for handling. It can be seen that this simulation can be carried out since the topology of the networks $N_i$ is identical to the topology of network $N_{i+1}$, except that we have to take care of one important difference: In network $N_i$ there is a virtual channel corresponding to edge $e_{i+1}^{u,v} = (p_u, p_v)$ between nodes $p_u$ and $p_v$ (which may be secure, authentic or tamperable depending on the adversary structure), while there is no such corresponding channel in network $N_{i+1}$. However, in protocol $\Pi^{i+1}$ when a message is to be sent on the channel $e_{i+1}^{u,v}$ the parties execute protocol $\gamma$ for message transmission, chosen from the family of of message transmission protocols $\gamma$, for transmitting message from node $p_u$ to node $p_v$ using only the infrastructure of incomplete network $N$.
We shall prove that $\mathcal{A}_i$ can carry out an (almost) perfect internal simulation of adversary program $\mathcal{A}_{i+1}$ taking into account this subtle but not negligible difference. The proof is by induction on the number of steps or super-rounds of execution of the protocol. We shall show that the following two conditions are simultaneously true by induction on the number of Steps of MPC protocol $\Pi^i, \Pi^{i+1}$:[9]

(a) *The distribution of views of $\mathcal{A}_{i+1}$, internally simulated by $\mathcal{A}_i$ attacking protocol $\Pi^i$'s execution on network $N_i$ is indistinguishable from the distribution of views of adversary $\mathcal{A}_{i+1}$ attacking protocol $\Pi^{i+1}$'s execution on network $N_{i+1}$, after every step.*

(b) *The distribution of the views of the honest parties $H_i$, generated from execution of protocol $\Pi^i$ is indistinguishable from the distribution of the views of the honest parties $H_{i+1}$ generated from execution of protocol $\Pi^{i+1}$, after every step.*

First, fix a vector of corruptions $\overrightarrow{C}$ by the adversary $\mathcal{A}_{i+1}$ for either execution. Now we shall proceed by induction on the number of steps $j$. We note here that the base case $j = 0$ has the same argument, albeit simplified, so we directly proceed with the induction hypothesis.
By induction hypothesis, assume $\mathbf{View}_{H_i,\Pi^i,j} \approx \mathbf{View}_{H_{i+1},\Pi^{i+1},j}$ and $\mathbf{View}_{\mathcal{A}_i,\Pi^i,j} \approx \mathbf{View}_{\mathcal{A}_{i+1},\Pi^{i+1},j}$ after Step $j$. Then, after Step $j+1$, $View_{H_i,\Pi^i,j+1} \approx View_{H_{i+1},\Pi^{i+1},j+1}$ and $View_{\mathcal{A},\Pi^i,j+1} \approx View_{\mathcal{A},\Pi^{i+1},j+1}$.
**Internal simulation of adversary $\mathcal{A}_{i+1}$ by $\mathcal{A}_i$ for step $j+1$:** For this we have the following four cases: (1) Honest parties sending messages to other honest parties (2) Honest parties sending messages to corrupted parties (3) Corrupted parties sending messages to honest parties (4) Corrupted parties sending messages to other corrupted parties (As stated in claim F we can assume that for adversary $A_{i+1}$ attacking protocol $\Pi^{i+1}$, the corrupted parties scheduled to initiate the process of sending their messages for this step first wait to receive all the messages sent by the honest parties in the first half of the Step/super-round and then send their messages during their slots in the latter part of the Step/super-round). We discuss each case separately:

(a) Honest party $p_u$ sending messages to another party $p_v$: We have four cases (a) Messages sent over the real channel in a single hop: In this case the view of the adversary is simply not affected (b) Message sent over virtual channels other then channel $e_{i+1}^{u,v}$: In this case the virtual channel could be uncorrupted, then the view of the adversary is unaffected. Else, the virtual channel could be (passively or actively) corrupted. In either case the adversary $A_i$ simply forwards the message to $A_{i+1}$. In the latter case, $A_{i+1}$ responds back with the message to be forwarded to the other honest

probability of error in the transmission translating to an overall at most negligible probability of error for the correctness of computation.

[9] The reader may recall the distinction on Step/super-round, phase, slots and rounds previously made, 19.

party in round $r-2$[10] which is then forwarded by $A_i$ to the virtual channel which in turn forwards it to party $p_v$. (c) Message sent by $p_u$ to $p_v$ by the execution of protocol $\gamma(p_u, p_v, N, s, c)$. Again this could be done (almost) securely or the execution results in passive corruption of the message by the adversary or active corruption. For the first case intermediate messages are received by the corrupted parties and these are forwarded by $A_i$ as it is to $A_{i+1}$, which subsequently forwards it back to $A_i$ so the message can be forwarded to the next honest party along whatever relevant path of nodes on the underlying network $N$. This results in neither $A_i$ nor $A_{i+1}$ learning anything intelligible from the execution of $\gamma$. In the second case $A_i$ and hence $A_{i+1}$ shall learn the message value at an appropriate round. In the third case, $A_{i+1}$ is in fact able to corrupt the message being sent from node $p_u$ to $p_v$ by the execution of protocol $\Pi^i$. Now, the way the internal simulation of $A_{i+1}$ by $A_i$ has been set up, $A_{i+1}$ has whatever maximum amount of knowledge it needs for further forwarding the corrupted message along the path at this round as it has during the execution of protocol $\Pi^{i+1}$.

Finally, we have case (d) When the message is sent from $p_u$ to $p_v$ over the virtual channel $e_{i+1}^{u,v}$ which is in network $N_{i+1}$ but not in network $N_i$. We note that this is the first case in which the protocol $\Pi^{i+1}$ differs from protocol $\Pi^i$ and for which simulation of $A_{i+1}$ may be non-trivial. For this case, recall that in protocol $\Pi^{i+1}$ message transmission protocol $\gamma(p_u, p_v, N, s, c)$ is executed between $p_u$ and $p_v$ utilizing the underlying infrastructure of the network $N$ to send the message. Note that for protocol $\Pi^i$ this transmission is taking place over the virtual channel $e_{i+1}^{u,v}$. We have the following three cases: $e_{i+1}^{u,v}$ is secure, $e_{i+1}^{u,v}$ is passively corrupted, or $e_{i+1}^{u,v}$ is actively corrupted. (1) When $e_{i+1}^{u,v}$ corresponds to a secure channel, then $A_i$ simply runs the simulator for protocol $\gamma(p_u, p_v, N, s, c)$ and since the transmitted message is hidden from the adversary in this case so the adversary $A_i$ also does not need any message to carry out a perfect simulation of adversary $A_{i+1}$'s view. (2) When $e_{i+1}^{u,v}$ is passively corrupted, then the adversary $A_i$ starts the simulation of $A_{i+1}$'s view by invoking the simulator for $\gamma(p_u, p_v, N, s, c)$ for this case, without possessing the message. Meanwhile, $A_{i+1}$ gets the message sent over the channel in the second round by $e_{i+1}^{u,v}$. It uses this message in appropriate round to complete the simulation of $\mathcal{A}_{i+1}$'s view for the execution of $\gamma(p_u, p_v, N, s, c)$. (3) Finally, for the last case $e_{i+1}^{u,v}$ is actively corrupted by the adversary $A_{i+1}$. In this case, $A_i$ starts the simulation of the view of $A_{i+1}$ for the execution of protocol $\gamma(p_u, p_v, N, s, c)$. Meanwhile, $A_i$ receives the signal from the virtual channel that it has received some message sent by the honest party $p_u$, while it continues to simulate $A_{i+1}$ for the first few steps of protocol $\gamma(p_u, p_v, N, s, c)$, till $A_{i+1}$ clears the intent whether it is going to corrupt the "plain" message or the message hidden by the one time pad. $A_i$ then specifies this intent to the virtual channel, which returns it with (1) message in clear which is then used to simulate the rest of the execution of protocol $\gamma(p_u, p_v, N, s, c)$ (2) message hidden under one time pad which is used to simulate the rest of the execution of protocol $\gamma(p_u, p_v, N, s, c)$. Whatever message (after corruption) is forwarded by $A_{i+1}$ in the simulation of $\gamma$ is then forwarded to the virtual channel in round $r-2$ when then forwards it to the appropriate sender in the next round (after XORRing the received message with the one time pad with which it initially hid the message if the case was of corruption with eavesdropping). We argue that the view of the adversary $A_{i+1}$, during the execution of protocol $\Pi^i$ is as enhanced and indistinguishable from the view that would have been constituted during the execution of $\Pi^{i+1}$ as necessary and sufficient to corrupt the message appropriately.

(b) Corrupted party $p_u$ sending a message to another node $p_v$: Note, that wlog we have assumed that corrupted parties send their messages during the second half of the phase after they have received/overheard whatever messages they could have, and which were sent by the honest parties i.e., by round $r$ all the messages sent by the honest parties have been received by the other relevant parties and the views of the adversary $A_i$, $A_{i+1}$ has been as updated as it could have been in this respect. The second lag of this step starts in round $r+1$ and terminates in final round $2*r+1$ of this step.

[10] We note here that because of the choice of our parameters by round $r-3$ of a step all the messages sent by the honest parties are received by the adversary much before this round. The adversary is then in a position to compute appropriate messages to be sent to other corrupted parties or honest parties based on this data.

The internal simulation of $A_{i+1}$ by $A_i$ is rather straightforward for this case as $A_i$ and hence $A_{i+1}$'s has already received all the requisite messages sent by the honest parties in the first half of this step and hence it has the appropriate view, that it would have while executing protocol $\Pi^{i+1}$, for $A_{i+1}$ to compute the messages that need to be forwarded over the individual channels/to the appropriate parties. $A_i$ extracts all the different messages that would be sent by $A_{i+1}$ by forward simulation (in just one computational phase of a single round $A_i$ can do this as it has the control over the clock of $A_{i+1}$. And, since the corrupted parties are the only parties that are scheduled to send messages in this part of the step, so the entire simulation of $A_{i+1}$ can be carried out without any participation by the honest parties) of a copy of $A_{i+1}$. $A_i$ then carries out the requisite simulation of adversary $A_{i+1}$ for this step, while sending all the extracted messages to appropriate destinations like virtual channels or real channels or through executions of protocol $\gamma(p_u, p_v, N, s, c)$.

Let us verify that the distribution of the views of the honest parties $H_{i+1} = H_i$ and the adversary $\mathcal{A}_{i+1}$ after execution of Step $j+1$ of protocol $\Pi^i$ on network $N_i$ and $\Pi^{i+1}$ on network $N_{i+1}$ are statistically indistinguishable[11], after the clean up at the end of Step $j+1$ follows from the correctness of construction of the simulations described above. In particular, by inductive hypothesis the messages chosen to be sent by the honest parties at the beginning of the first phase of the Step $j+1$ have identical (with negligible difference) distribution for the two cases of protocol $\Pi^i$'s execution on $N_i$ and $\Pi^{i+1}$'s execution on network $N_{i+1}$. Now by simulatability of $\gamma$ the view of $\mathcal{A}_{i+1}$ are updated indistinguishably for this phase (for all the three types of channels $e^{u,v}_{i+1}$) for the two cases.

For the second phase of Step $j+1$ the participation of the honest parties is only passive i.e., the honest parties are not involved in sending any non-trivial messages i.e., the role of the honest parties is of dummies and in fact can be simulated trivially. Since only corrupted parties send messages during the second phase of the protocol $\Pi^{i+1}$, the messages to be sent by the corrupted parties are not influenced by any other external factors except the corrupted parties themselves. Since the views of $\mathcal{A}_{i+1}$ are identically distributed at the end of the first phase of the Step $j+1$, and all participation by honest parties may as well be seen as participation by dummies, it is easily seen from the above description of protocol $\Pi^{i+1}$ that the messages chosen to be sent by the corrupted parties controlled by $\mathcal{A}_{i+1}$ are identically distributed and identically influenced for the two cases: When protocol $\Pi^i$ is executed on network $N_i$ and when protocol $\Pi^{i+1}$'s executed on network $N_{i+1}$, after transmission on each of the channels the views of $\mathcal{A}_{i+1}$ are identically distributed for the two cases.

Finally, after the cleaning up phase of the Step $j+1$, which involves deleting all intermediate garbage messages etc., it follows from the above argument that the views of the honest parties are also identically distributed for the two cases, when protocol $\Pi^i$ is executed on network $N^i$ or when protocol $\Pi^{i+1}$ is executed on network $N^{i+1}$. This implies that the distribution of the views of the unsacrificed honest parties at the termination of the input commitment phase formed for the two cases are indistinguishable. Furthermore, if honest parties start with the same initial inputs then they commit to the same input values for both the cases. In particular, $reveal_{\Pi^{i+1}} = reveal_{\Pi^i}$ (Given these facts, it is easy to verify the correctness of the input commitment phase of $\Pi^{i+1}$). The correctness of the computation phase follows from the same reasoning that the distribution of the views of the unsacrificed honest parties are same for the two cases. In particular, conditioned on the fact that the parties commit to the same input values the output value for the two cases is same, which completes the proof of correctness of $\Pi^{i+1}$.

This completes the inductive hypothesis for the correctness.

2. Privacy: For this we demonstrate a simulator that simulates the views of $\mathcal{A}_{i+1}$, attacking $\Pi^{i+1}$'s execution on network $N_{i+1}$, such that the distribution of views of $\mathcal{A}_{i+1}$ generated by the simulator is indistinguishable from the distribution of the views of $\mathcal{A}_{i+1}$ generated from real execution of protocol $\Pi^{i+1}$.

[11] Note that the last step of the protocol $\gamma$ is a **clean up** phase in which all honest parties delete all intermediate messages exchanged, transmitted, forwarded etc. during the execution of protocol $\gamma$. Now semi-honest parties may not necessarily delete such values but they are not supposed to consider these values as part of their views on the basis of which do further computations - which is a requisite for their "honest" looking behavior. Thus, for all the purpose of discussing distribution of views of the honest parties, we may just consider that after the execution of $\gamma$ phase - the clean up phase deletes all the intermediate messages

We shall employ induction hypothesis for the following adversary $\mathcal{A}_i$, as described in the Correctness condition proved above, that attacks $\Pi^i$, while internally simulating $\mathcal{A}_{i+1}$. By induction hypothesis, there exists a simulator $Sim_i$ that takes as input the identities, input and output values of the parties in subset $P - H_i$, adversary program $\mathcal{A}_i$, network $N_i$, $\overrightarrow{C}_i$ and generates a view (or more specifically distribution of views) of $\mathcal{A}_i$ that is indistinguishable from the view (or specifically distribution of views) of $\mathcal{A}_i$ generated from real execution of $\Pi^i$ on network $N_i$ with same parameters. Let $Sim_i$ be such a simulator.
Simulator $Sim_{i+1}$ for $N_{i+1}, \Pi^{i+1}, \mathcal{A}_{i+1}$ simulates adversary $\mathcal{A}_{i+1}$ as follows: First note that the simulator $Sim_{i+1}$ is given the same set of input and output values of party set $P - H_{i+1} = P - H_i$ as $Sim_i$ (except that $\overrightarrow{C}_i$ may contain the virtual channel $(p_u, p_v)$ unlike $\overrightarrow{C}_{i+1}$).
In general at the end of an execution, adversary $\mathcal{A}_i$ outputs its input tape and read tapes as the output and this is considered as the output of the adversary. In the adversary program $\mathcal{A}_i$ described above, $\mathcal{A}_i$ (indistinguishably) internally simulates $\mathcal{A}_{i+1}$ for generating the messages to be sent to the honest parties etc.. We make another enhancement to this adversary program: On termination $\mathcal{A}_i$ outputs, whatever is outputted by $\mathcal{A}_{i+1}$ (instead of its own read tapes, input tapes). Since, we have shown that in the proof of the correctness condition above, $\mathcal{A}_i$ perfectly simulates the view of $\mathcal{A}_{i+1}$ and, hence simulator $Sim_i$ is going to output the view of $\mathcal{A}_{i+1}$, generated in the process, whose distribution will be perfectly indistinguishable from the distribution of views of $\mathcal{A}^{i+1}$ generated from real execution of $\Pi^{i+1}$. (The picture looks like this: Earlier $Sim_i \leftrightarrow \mathcal{A}_i(\mathcal{A}_{i+1})$ and now $Sim_{i+1} = Sim_i(\mathcal{A}_i) \leftrightarrow \mathcal{A}_{i+1}$.)
The proof of correctness of simulator $Sim_{i+1}$ follows from the proof of correctness of simulator $Sim_i$, and the proof of correctness of internal simulation of adversary $\mathcal{A}_{i+1}$ by $\mathcal{A}_i$ given above, otherwise it would contradict the correctness of either of them.

This proves the induction hypothesis.

The I. H. can be invoked till there exists a virtual channel in the network that can be replaced by a simulated channel. So, successive applications of I. H. results in network $N_l = N$ which is the original incomplete network $N$ under consideration.

Putting together with the base case, we have that if $N$ is a network, that possesses $\mathcal{T}_{\beta_{N,c}}$-Communicability Property, 10, then there exists a multiparty computation protocol $\Pi_N$, to be executed on $N$, that $\mathcal{T}$-securely evaluates function $f$, as according to Definition 14.

*Remark 7.* Probabilistic functions can be handled in similar manner as deterministic functions with minor enhancements/considerations. For the case of probabilistic functions, the simulator keeps with it the set of output values that belong to the equivalence class corresponding to the output value of the function. Based on the random choices made by it during the simulation while simulating the honest and sacrificed parties or by the messages received by it from the parties corrupted by the adversary this subset of possible output values may get reduced. The simulator takes this into account and works with the reduced sized equivalence class. The rest of the simulation proceeds as it is.

## G Construction of networks when even the honest parties are also honest-but-curious

In [Vay06], I showed construction of networks that provide security against hones-but-curious parties given that the original network may not. This is not required for any of the networks for we achieve almost everywhere secure multiparty computation but may be useful for other networks. [Note that in [Vay06] this construction was used for networks considered by Upfal, [Upf92], however I observed that even for LPS expanders considered by Upfal the construction is not necessary as [BBC$^+$06] observed a general result from Upfal's theorem: For any expander if a small linear fraction of nodes is deleted, then the resulting network is still has a (smaller sized) subnetwork which has good enough expansion properties. For LPS expanders, the subsets of nodes for which Upfal, [Upf92], guarantees pairwise authentic channels (just like [DPPU88]) is exactly this subnetwork. Thus, several node disjoint paths of length $c * \lg_2 n$ are guaranteed between every pair of nodes of this subnetwork, all of which are also non-faulty and so this construction

is not needed. Nevertheless, we present here a way of constructing networks which for which just a single path of non-faulty nodes is guaranteed to construct networks for which at leat two node disjoint paths of non-faulty nodes will be ensured, at the cost of tolerating slightly lesser (a constant fraction) number of corruptions.

The following lemma is taken verbatim from [Vay06] but can be adjusted to our setting of $\mathcal{T}$ adversaries.

**Lemma 1.** *Let $N = (V,E)$ be a network with n nodes. Let $\Pi$ be a protocol such that for t-restricted adversaries there exists a subset $S \in V$, $|S| \geq n - f(t)$ such that, $\forall u,v \in S$:*

1. *$\Pi$ simulates a public reliable channel between nodes u and v;*
2. *there exists a secure path of non-faulty nodes of length $O(\lg_2 n)$ between nodes u and v.*

*Then there exists a network $N' = (V',E')$, $|V'| = 2*n$, and a protocol $\Pi'$ such that for t-restricted adversaries there exists a subset $S \in V$, $|S| \geq n - f(t)$ such that, $\forall u,v \in S$:*

1. *$\Pi'$ simulates a public reliable channel between nodes u and v;*
2. *there exists at least two node-disjoint paths of non-faulty nodes of length $O(\lg_2 n)$ between u and v;*
3. *$|S'| \geq 2*n - O(f(t))$.*

*Proof.* Network $N'$ with $2*n$ is constructed as follows: Take two copies of network $N$ of size $n$ each. Call them $N_1$ and $N_2$. Define $V' = V_1 \bigcup V_2$ and add edges between the isomorphic nodes of network $N_1$ and $N_2$. The resulting network is called $N'$.

Both sub-networks, $N_1$ and $N_2$, satisfy the premises of the theorem.

Let $T_1$ be the set of malicious nodes in $N_1$ and let $Iso(T_1)$ be the set of nodes in $N_2$ that is isomorphic to set $T_1$. Correspondingly, define sets $T_2$, $Iso(T_2)$ for subnetwork $N_2$. Let $T' = T_1 \bigcup T_2$.

Let $S_1$ be the subset of nodes in $N_1$ such that, $\forall : u,v \in S_1$, protocol $\Pi$ simulates a reliable channel between $u$ and $v$ and there exists a path of length $O(\lg_2 n)$ between $u$ and $v$. Let subset $S_2$ be defined analogously for network $N_2$.

Now, $\forall u,v \in S_1 - Iso(T_2) \bigcup S_2 - Iso(T_1)$, $u \neq v$, we show that a public reliable channel and two node-disjoint paths of non-faulty nodes of length $O(\lg_2 n)$ exists between nodes $u$, $v$. We have the following cases:

1. $u,v \in S_1 - Iso(T_2)$ : A public reliable channel and a path of non-faulty nodes of length $O(\lg_2 n)$ nodes between nodes $u$ and $v$ is given to us by the premises of the theorem.
   The second path is as follows: 1) $u \rightarrow u'$, where $u'$ is the node in $N_2$ isomorphic to node $u$ in $N_1$, connected by an edge in $E'$. 2) $u' \rightarrow v'$ :, where $v'$ is the node in $N_2$ isomorphic to node $v$, connected by a path of non-faulty nodes of length $O(\lg_2 n)$, and 3) $v' \rightarrow v$, the edge in $E'$.
2. $u,v \in S_2 - Iso(T_1)$ : Similar to case 1.
3. $u \in S_1 - Iso(T_2)$, $v \in S_2 - Iso(T_1)$ : Let $v'$ be the node in $N_1$ that is isomorphic to node $v$. Using protocol $\Pi$ we can simulate a reliable channel between nodes $u$ and $v'$. Next, the message received by $v'$ through this channel is forwarded to node $v$. The two paths of non-faulty nodes of length $O(\lg_2 n)$ are as follows.
   (a) $u \rightarrow u'$, where $u'$ in $N_2$ is isomorphic to $u$, followed by $u' \rightsquigarrow v$: a path of non-faulty nodes of length $O(\lg_2 n)$.
   (b) $u \rightsquigarrow v'$, where $v'$ is the node isomorphic to $v$ in $N_1$, followed by $v' \rightarrow v$.
4. $u \in S_2 - Iso(T_1)$, $v \in S_1 - Iso(T_2)$ : Similar to case 3.

Let us estimate the size of the subset $S' = S_1 - Iso(T_2) \bigcup S_2 - Iso(T_1)$:
$|S'| \geq n - f(t_1) - f(t_2) + n - f(t_2) - f(t_1) = 2*n - O(f(t))$, as $t_1, t_2 \leq t$ and $f$ is a monotonically increasing function.